

Чек-лист: как готовить LLM-сервис к adaptive compute

Отдельный PDF для [downloads/checklist.pdf](#). Его можно приложить к статье как практическую памятку для команды разработки, product owner и ML-инженера.

Проверка статуса claims

- Не писать, что DeepSeek внедрила energy-aware inference, пока нет официальной документации, кода или benchmark'ов.
- Разделять GitHub-сигнал, research proposal, API-документацию и production-релиз.
- Для энергетических заявлений требовать методику измерения, baseline и аппаратную конфигурацию.
- Перед обновлением статьи сверять pricing-страницы DeepSeek, потому что условия API могут меняться.

Метрики и наблюдаемость

- Логировать input tokens, output tokens, latency до первого токена и полную задержку.
- Отдельно фиксировать выбранный reasoning-режим, retry rate, ошибки по таймауту и стоимость вызова.
- Для агентных сценариев считать всю цепочку: планирование, tool calls, retries, проверки и финальный ответ.
- Хранить бизнесовую оценку качества: решена задача или нет, была ли эскалация, прошли ли тесты.

Маршрутизация запросов

- Разделять запросы на простые, спорные, сложные и критичные до обращения к модели.
- Не отправлять все задачи в максимальный reasoning-профиль по умолчанию.
- Для простых задач использовать более дешевый режим, если качество остается достаточным.
- Для сложных и критичных задач включать более сильный профиль или дополнительную проверку.

Fallback и контроль деградации

- Описать, когда дешевый режим должен передавать задачу в более дорогой профиль.
- Проверять экономию вместе с качеством, latency и долей ошибок.
- Тестировать редкие, но дорогие ошибки отдельно, а не только среднюю стоимость.
- Не считать adaptive compute успешным, если падает ключевая бизнес-метрика.