

Конкретнее на примере — начнём с того, что у вас есть версия API 1.0.

1. Вы добавили некритичные изменения (добавление новых эндпоинтов, добавление новых необязательных полей в входные данные и добавление новых полей в выходные данные). После, по правилам семантичности версий, увеличили цифру в версии API — 1.1. Клиенты могут переходить на эту версию API без проблем, так как нет критичных изменений.(заметьте, у вас пока две активные версии API — 1.0 и 1.1.)
2. Далее вы добавили некритичные изменения и уведомляете клиентов о выпуске API 1.2 и скором отключении 1.0.
3. Таким образом, через некоторое время клиенты переходят на 1.1 (или сразу на 1.2), и у вас только две активные версии API — 1.1 и 1.2. (1.0 вы отключили)
4. Далее вы решили добавить критичные изменения (изменение или удаление эндпоинтов, изменение или удаление полей данных, ИЗМЕНЕНИЯ В ОТВЕТАХ ОШИБКАХ И КОДОВ СТАТУСОВ). Вы готовы выпускать версию API 2.0 и уведомляете клиентов о выпуске 2.0 и скором отключении 1.1, и просите перейти всех на активные API (1.2 или сразу на 2.0)
5. и т. д. по такому же принципу

Таким образом, вы создаете фундамент для критичных изменений из релизов API с некритичными изменениями. У клиентов есть время адаптироваться и подключиться к вашим новым версиям API. Для некритичных изменений им не нужно сильно менять код, а для критичных — нужно.

Запомните

Не являются критичными изменениями:

- добавление новых эндпоинтов,
- добавление новых необязательных полей в входные данные,
- добавление новых полей в выходные данные.