MAKE THINGS

Hugging Face, End to End

Read a model card licence-first, run a Space for free, keep a token safe, publish something of your own.

First edition, September 2026

Addaly

Series editor: Dr Mustafa Mun

Draft, open for academic review

MAKE THINGS

Hugging Face, End to End

Read a model card licence-first, run a Space for free, keep a token safe, publish something of your own.

Series editor: Dr Mustafa Mun

Addaly

First edition, September 2026 · Draft, open for academic review

Hugging Face, End to End

First edition, September 2026, build 62a26075.

Written by AI models to a written standard, with Dr Mustafa Mun as series editor. Exam keys checked blind by a second model: 3,665 of 3,666 agreed. Academic review invited.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

How to cite: *Mun, M. (2026). Hugging Face, End to End. Addaly. addaly.com/learn/hugging-face.*

This book is the printed form of the course of the same name at addaly.com/learn/hugging-face. The website is the living version and is corrected first. Where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. The answers to every check, test and examination question are at the back.

8 modules · 75 lessons · 28 figures · 8 case studies · 55,572 words · about 11 hours

Brief contents

1

MODULE 1

The Hub, read properly

Find a specific model on the Hub using filters and the Python API rather than the search box, read its repository files to say what architecture it is, what format its weights are in and what code will run when you load it, and explain what the download counter is actually counting.

Lesson 1 · 10 min

The Hub is git, and that explains everything else

THE ONE THING TO KEEP

A model on the Hub is a git repository with big files stored beside it, so it has versions, history and pull requests — and if you do not pin a revision, the model under your code can change without your code changing.

Most people use the Hub as a download button

Open a model page — `openai/whisper-large-v3`, say — and look at the tabs across the top: Model card, Files and versions, Community. Then look inside Files and versions. There is a commit history. There are branches. Under Community there are pull requests sitting alongside the discussion threads.

None of that is decoration. A Hugging Face repository **is** a git repository. The model card is a file called `README.md` inside it. The weights are files committed to it. The Hub is a git host that happens to be full of neural networks, and almost everything confusing about the platform stops being confusing once you hold that one fact.

What being git actually buys you

A model has versions, and `main` moves. When a lab fixes a tokenizer bug or reuploads a corrected checkpoint, the repository gets a new commit. If your code says `from_pretrained("some-org/some-model")`, it means *whatever is on main at the moment it runs*. That is fine while you are experimenting and a real hazard in something you have shipped, because your model can change without your code changing. Every loading function takes a `revision` argument, and it accepts a branch, a tag, or a commit hash:

```
from transformers import AutoModel

model = AutoModel.from_pretrained(
    "sentence-transformers/all-MiniLM-L6-v2",
    revision="PASTE_THE_COMMIT_HASH_FROM_THE_HISTORY_TAB",
)
```

Copy the hash from the History link in Files and versions. Pinning costs you nothing and removes a whole category of Monday morning.

Figure 1.1

Eighteen months on one model's main branch

- **`Day 0`**
  First release: weights, tokenizer, and a card with no licence field at all.
- **`Week 2`**
  A contributor's pull request adds the licence tag and a safetensors copy of the weights.
- **`Month 3`**
  The chat template inside tokenizer_config.json is corrected. Every prompt rendered before this date was subtly the wrong shape.
- **`Month 7`**
  A reuploaded checkpoint fixes a tokenizer bug. Same repository, same name, different bytes.
- **`Month 11`**
  The repository is gated retrospectively. Unauthenticated downloads start returning 403 in a pipeline nobody touched.
- **`Month 18`**
  A second contributor adds a stated limitation to the card: three lines that save the next reader a week.

None of these commits changed the model's name. Code that did not pin a revision picked up every one of them, the next time it ran, with no change of its own. The hash in the History tab is what makes a result reproducible.

You can read the diff. History shows what changed and when. If a model started behaving differently last week, that is the first place to look, and often the last.

Anyone can open a pull request. Someone adds a missing licence tag, converts weights to safetensors, translates a card into Hindi. On the Hub these live as real git refs like `refs/pr/4`, so you can download and test a proposed change before it is merged.

Nothing has to be public. A repository can be private, and every command below works the same way with a token.

The big files are not really in git

Git was built for source code. It handles a 5 GB tensor file badly. So the Hub keeps the *pointers* in git and the *bytes* in a separate large-file store — Git LFS originally, and since 2025 Hugging Face has been moving repositories onto its own chunk-level storage that deduplicates blocks, so reuploading a slightly changed checkpoint does not resend the whole thing. You do not need to know which one sits behind a given repo. You do need to know one consequence.

Run this on a machine that does not have Git LFS installed:

```
git clone https://huggingface.co/openai/whisper-small
```

You get a folder with all the right filenames and a `model.safetensors` that is about 130 bytes. It is not corrupt and the download did not fail. It is the pointer file — a few lines of text naming an object hash — because git faithfully cloned exactly what was committed, and nothing fetched the real object. People lose an hour to this and conclude the model is broken.

Three ways to actually get the files

The browser. Every file has a download link. On a phone this is the only route, and for one 200 MB file it is fine. Direct URLs follow a single pattern worth memorising: `https://huggingface.co/<owner>/<repo>/resolve/main/<filename>`.

The command line. Install the client and use it instead of raw git:

```
pip install -U huggingface_hub
hf download openai/whisper-small
```

The command was called `huggingface-cli` for years and was renamed to `hf` in 2025. If your shell says `hf: command not found`, upgrade the package, or use the old name — both are common in tutorials you will find. `hf download` resolves large files properly, resumes an interrupted transfer, and puts everything in a shared cache rather than a folder you will forget about. That last part matters more than it sounds; lesson seven is about the disk it fills.

Python, when you want one file. A repository often holds three copies of the same weights in different formats. You rarely want all of them:

```
from huggingface_hub import hf_hub_download

path = hf_hub_download("openai/whisper-small",
"model.safetensors")
```

For a subset, `snapshot_download` takes `allow_patterns` — passing `[".safetensors", ".json"]` skips the duplicate PyTorch `.bin` files and can halve what you pull down.

Do this now

Pick any model you have used through somebody else's app. Open Files and versions, then History. Count how many times it has changed since it was announced, and read the message on the most recent commit. That commit is the version you have been using.

BEFORE YOU MOVE ON

Someone runs `git clone` on a model repository and ends up with a `model.safetensors` of about 130 bytes containing a few lines of text. What has happened?

A. The repository is gated and they were not authenticated, so the server returned a placeholder file instead of the weights.

B. The transfer was cut short by the network and needs to be resumed before the file is complete.

C. Git cloned the pointer file that stands in for the weights; the actual bytes live in a large-file store that Git LFS or `hf download` fetches separately.

D. The branch `main` had moved to a commit made before the weights were uploaded, so only the metadata existed at that revision.

The answer and why: p. 441

Lesson 2 · 9 min

The search box is the worst way in

THE ONE THING TO KEEP

Hub search matches repository names and card text rather than meaning, so the filter sidebar — task tag, library, licence, and the base-model links to fine-tunes and quantizations — is the real interface, and the sort order answers popularity rather than quality.

Typing what you want does not work

Go to the Hub and type `sentiment analysis` into the search box. You get repositories whose *names* contain those words. You do not get `cardiffnlp/twitter-roberta-base-sentiment-latest`, which is one of the

most used sentiment models on the platform, unless the substring happens to match. Hub search is largely a text match over repository names and card content. It is not a semantic search over what models do.

This catches everyone once. The model you are looking for exists, has been downloaded four million times, and does not appear, because the person who uploaded it called it `distilbert-base-uncased-finetuned-sst-2-english` and never wrote the phrase you typed.

The filters are the real interface. The search box is for when you already know the name.

The filter that does most of the work

Every model repository carries a `pipeline_tag` in its card metadata — one value from a fixed list: `text-classification`, `automatic-speech-recognition`, `image-segmentation`, `text-generation`, and around fifty others. That tag is what the **Tasks** sidebar filters on, and unlike free text it is a controlled vocabulary. Pick the task, and you are looking at models that claim to do that job and nothing else.

The filters compose into the URL, which is worth knowing because it lets you keep a search as a bookmark or paste it to a colleague:

```
https://huggingface.co/models?pipeline_tag=automatic-speech-recognition&library=transformers&language=hi&sort=downloads
```

That one reads: speech recognition, loadable with transformers, tagged Hindi, most downloaded first. Four constraints, one line, and it narrows a million repositories to a screenful.

The filters worth reaching for, in the order they usually help:

- **Tasks** — the controlled vocabulary above.
- **Libraries** — `transformers`, `sentence-transformers`, `diffusers`, `gguf`, `onnx`. This is a filter on *what will load it*, which is often the real question.
- **Languages** — tagged by the uploader, so it is a claim rather than a measurement.

- **Licences** — `apache-2.0`, `mit`, `cc-by-nc-4.0`, and the bespoke ones. Filter here early if the work is commercial.
- **Other** — model size ranges, and the `base_model` relationships described below.

Sorting: trending is not quality

The sort control offers Trending, Most likes, Most downloads, Recently created and Recently updated. Each answers a different question, and none answers "which is best".

- **Most downloads** is a thirty-day window, and it favours models that libraries load by default. It tells you something is safe and widely exercised. It tells you nothing about whether it suits your language or your text length.
- **Trending** weights recent activity, so it surfaces last week's release. Useful for knowing what happened while you were on holiday, useless for picking a production dependency.
- **Recently updated** is the one people forget, and it is how you find out whether a model still has a maintainer.

Finding the variants of a model you already have

This is the trick that saves the most time. A model card declares its parent with a `base_model` field, and the Hub indexes that relationship in both directions. On any base model's page there is a line reading something like *"Finetunes: 3,417 · Quantizations: 891 · Adapters: 2,104"*, and each is a link.

So when you have found `meta-llama/Llama-3.1-8B-Instruct` and your laptop cannot run it at full precision, you do not search for "llama 8b small". You click **Quantizations** and get every GGUF, AWQ and GPTQ conversion anyone has published, in one filtered list. The same route finds the adapter someone already trained for your language, or the fine-tune already specialised for medical text.

The dataset and Space filters are the same idea

Datasets filter by task, size category, language, licence and format. The size filter is a band — `1K<n<10K`, `10M<n<100M` — and it is populated automatically from the dataset viewer rather than claimed by the uploader, which makes it one of the more trustworthy facets on the site.

Spaces filter by SDK (`gradio`, `streamlit`, `docker`, `static`) and by hardware. Filtering Spaces to `gradio` and sorting by likes is the fastest way to find a working reference implementation of almost any task, because a Space that runs is a configuration that works.

Full-text search, when you need it

There is a full-text option that searches inside model cards rather than only names. It is reachable from the search results page and it is genuinely useful for phrases that only ever appear in prose — a benchmark name, a paper title, a specific dataset. Searching card text for `WER` and filtering to speech recognition finds the models whose authors actually reported an error rate.

Its limitation is the same one that afflicts the whole site: a card is a claim written by the uploader. Searching for a phrase finds people who wrote the phrase, not people whose model is good at the thing.

Do this now

Pick a task you care about. Build the URL by hand with three filters — task, library, licence — and sort by downloads. Then open the third result rather than the first, and look at when it was last updated. The gap between the ranking and the maintenance is the thing most people never check.

BEFORE YOU MOVE ON

A team needs a 4-bit version of a model they have already chosen so it fits on a laptop. What is the reliable way to find one on the Hub?

A. Search for the model name plus a term like `4bit` or `quantized`, since conversions almost always put that in the repository name.

B. Open the base model's page and follow its Quantizations link, which indexes every published conversion.

C. Filter the models list to the `gguf` library and sort by downloads, then pick the highest-ranked result.

D. Use full-text card search for the model name, because conversions cite their parent in the card prose.

The answer and why: p. 442

Lesson 3 · 11 min

Read the licence before you read the demo

THE ONE THING TO KEEP

Check the licence family and the base model before anything else, because a fine-tune inherits its parent's terms no matter what its own card claims.

The order everyone reads it in is wrong

Most people open a model page, look at the sample outputs, decide they like it, and start building. The licence gets checked at the end, if at all — usually the week before launch, by someone who is now unhappy.

A model card is a `README.md` with a YAML block at the top. That block carries the machine-readable facts: `license`, `base_model`, `datasets`, `pipeline_tag`, `language`, `tags`. The prose underneath is written by whoever uploaded the

CASE STUDY · MODULE 1

The name that was spelled two ways

An illustrative case, built from documented patterns.

A legal aid clinic in Pune transcribing Marathi and Hindi hearing audio into case files, with four volunteers and twenty donated GPU hours a week.

Since January the clinic had been running one script. It took an audio file from a hearing, pushed it through a large open speech model, and wrote a transcript into the matter's folder. Fourteen hundred hours of audio had gone through it by August. The script was ninety lines long and nobody had touched it since the week it was written.

Nobody in the clinic thought of it as software. It was the thing that turned nine hours of a bad recording made on a phone at the back of a courtroom into text a lawyer could read on a train. The volunteers called it the transcriber, and the only thing anybody ever changed about it was the folder it wrote to.

In August a paralegal called Sushma noticed that a respondent's surname appeared as Waghmare in one transcript and Vaghmare in another, in the same matter, from two hearings six weeks apart. She assumed a typing error until she found three more, all in files produced after June.

Anup, the volunteer who had written the script, opened the model's Files and versions tab and then its History. There were two commits since January. One in April replaced the tokenizer files. One in June reuploaded a corrected checkpoint, with a message saying it fixed a bug affecting proper nouns.

The script said `from_pretrained("openai/whisper-large-v3")`. No revision. So it had never meant one model. It had meant whatever was on the main branch at the moment each transcript was made, and the archive was not one archive but three, with nothing in any filename to say which.

The decision

The clinic quotes these transcripts in written submissions. Two options were on the table and both cost something real.

Re-transcribe everything. The donated allocation was twenty hours a week on a college lab machine. Fourteen hundred hours of audio at roughly real time is about three weeks of that allocation, during which no new intake could be transcribed. Intake does not pause. Three weeks meant roughly ninety new matters waiting on a transcript before anybody could read them.

Or pin from today, accept an archive produced by three different models, and carry the risk. That risk is not a daily cost. It is a credibility cost that arrives once, unpredictably, in front of a bench, when opposing counsel asks why two documents from the same clinic spell the same person's name differently.

Anup made one more check before anybody chose. The June commit message said the fix concerned proper nouns. That narrowed the problem from everywhere to one place, and one place happened to be what a legal transcript is mostly about. He also asked a question nobody had asked yet: how much of the archive belongs to matters still before a court. The answer was a hundred and ninety hours. The rest was closed, or kept for research.

The cost on one side was ninety delayed matters. The cost on the other was a document nobody could vouch for. The information that made the choice tractable was already sitting in the repository, free, in a commit message and a list of dates.

The clinic also had to decide what it was willing to say out loud. A transcript in a submission is offered as a record of what was said. Saying that some of the clinic's transcripts were produced by a model that has since been corrected is an uncomfortable sentence, and it is a sentence somebody would eventually have to say either way — the only question was whether the clinic said it first, in a note it had written calmly, or second, in an answer to a question from a bench.

Anup pointed out one more thing that made the pinning decision easy even if the re-running decision was not. Pinning costs nothing. It is one extra argument in three function calls, it can be done in ten minutes, and it does not compete with anything for the donated GPU hours. Whatever they chose about the existing fourteen hundred hours, there was no version of the argument in which they should keep loading an unpinned model tomorrow.

YOUR ANSWERS

1. Anup's script had not been edited since January, yet the transcripts it produced changed. Explain the mechanism.

2. Reading the June commit message changed the size of the problem. Why is that, and what does it say about the History tab?

3. They re-ran a hundred and ninety hours rather than fourteen hundred. What makes that a defensible decision rather than a convenient shortcut?

STOP HERE

Write your answers before you turn the page. How it played out starts on p. 54.

CASE STUDY · MODULE 1

How it played out

They pinned. Every loading call in the script got a `revision` argument carrying the June commit hash, copied out of the History tab, including the one that loads the processor. They re-ran the hundred and ninety hours belonging to open matters, which took four days rather than three weeks, and left the closed archive as it was with a one-page note in the shared drive recording which date ranges came from which revision and what the June fix had changed.

Every transcript produced from then on carried a header line with the model identifier, the revision hash and the date it was made. Six months later a bench did ask about a spelling. The clinic answered in one sentence and produced the note. Writing the note had cost an afternoon.

The questions, discussed

1. Anup's script had not been edited since January, yet the transcripts it produced changed. Explain the mechanism.

A Hugging Face repository is a git repository and `from_pretrained("owner/name")` without a `revision` argument resolves to whatever is on the main branch at the moment the code runs. Two commits landed between January and August, so the same ninety lines of code loaded three different sets of weights and tokenizer files over the period. Pinning `revision` to a commit hash removes the whole category, and costs nothing.

2. Reading the June commit message changed the size of the problem. Why is that, and what does it say about the History tab?

The message said the fix affected proper nouns. That turned an unbounded worry — every word of fourteen hundred hours might differ — into a bounded one concentrated in names. History is a diff and not merely a list of dates, so it answers what changed as well as when. For a model that started behaving differently last week, it is the first place to look and often the last.

3. They re-ran a hundred and ninety hours rather than fourteen hundred. What makes that a defensible decision rather than a convenient shortcut?

They split the archive along the axis where the risk actually sits. A transcript from a closed matter is not going to be read back to a judge; one from an open matter might be. They also recorded what they had not re-run and why, which is the part that turns a scoped decision into an auditable one. A shortcut hides the gap; this documented it.

MODULE 1 TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record. The answers, and the reason behind each, are in the key on p. 440. If two go wrong, re-read the module before the exam.

1 A transcription script has not been edited since January, yet the transcripts it produced in July differ from those it produced in March. What is the mechanism?

A. A cached copy expired, so the weights were fetched again in a different precision.

B. from_pretrained without a revision loads whatever is on the main branch that day.

C. The Hub serves different mirrors to different regions and they drift apart over time.

D. Downloading a model twice gives different bytes, because large files are deduplicated.

2 You clone a model repository with plain git and find that model.safetensors is about 130 bytes. What has happened?

A. The repository is gated and the Hub served a small placeholder instead of the real weights.

B. The maintainer committed an empty file and pushed the real weights to a branch.

C. Git cloned the pointer; the bytes live in large-file storage and were never fetched.

D. The model is quantized to four bits, which is why the file is so small.

3 A model shows four million downloads over thirty days. What is the strongest inference you can draw from that number?

A. Enough different code paths have loaded it that obvious breakages are known.

B. It is the best-performing model available for its task at this size, by some margin.

C. Roughly four million distinct people have used it during the last month.

D. It is actively maintained, because a stale model stops being downloaded entirely.

4 Your laptop cannot run a model at full precision. What is the fastest route to a version that fits?

A. Type the model's name and the word small into the search box.

B. Sort the task by trending and take the most recent release.

C. Open the base model's page and follow its Quantizations link.

D. Filter the whole Hub by model size band and read through every result.

5 A card tags a fine-tune apache-2.0. Its base_model field names a repository under a non-commercial research licence. Which terms govern your paid product?

A. Apache 2.0, because a fine-tune is a new work and carries its own licence.

B. Whichever is the more recent, since licences are versioned like code.

C. Neither, until the uploader answers a question in the discussions tab.

D. The non-commercial terms, because a derivative inherits its base's terms.

6 A repository ships only safetensors weights, and its usage snippet passes trust_remote_code=True. What risk remains?

A. None, because safetensors contains no instruction stream and cannot execute code.

B. Python from the repository is downloaded and imported when the model loads.

C. The safetensors header can be crafted to call a function while it is being parsed.

D. Loading will fail, because that flag only applies to pickle checkpoints.

The ratio, and why it is not the same for everyone

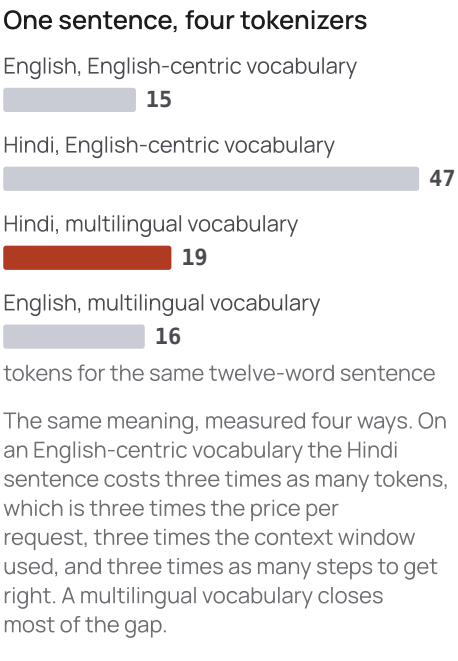
For English prose with an English-centric vocabulary, the working figure is about **1.3 tokens per word**, or roughly four characters per token. That number is where the rule of thumb "750 words ≈ 1,000 tokens" comes from.

It does not transfer. The same text in Hindi, Bengali, Tamil, Telugu or Marathi, tokenized by a model whose vocabulary was built mostly from English text, commonly runs two to four times as many tokens for the same meaning, because Devanagari and other Indic scripts fall back to multi-byte pieces. Try it:

```
for s in ["Where is my parcel?", "□□□□ □□□□□□ □□□□ □□?"]:
    print(len(tok(s)["input_ids"]), s)
```

Three consequences, all practical:

Figure 2.1



1. **Cost.** Per-token pricing means the same question costs more in Hindi than in English on the same model. This is a real, measurable inequity in how these systems are priced, and it is a property of the vocabulary rather than of the language.
2. **Context.** A 4,096-token window holds much less Hindi than English.
3. **Quality.** More tokens per word means the model has more steps to get right for the same content.

Models with multilingual vocabularies — Qwen, Gemma, Aya, IndicBERT and the muril family — narrow the gap substantially. Checking the ratio on your own text is a ten-second test that should inform model choice and almost never does.

Special tokens appear without being asked for

```
tok = AutoTokenizer.from_pretrained("bert-base-uncased")
print(tok("hello")["input_ids"])              # [101, 7592, 102]
print(tok.convert_ids_to_tokens([101, 102]))  # ['[CLS]',
'[SEP]']
```

You wrote one word and got three integers. Encoder models wrap input in markers; decoder models often prepend a beginning-of-sequence token. `add_special_tokens=False` turns this off, which you want when you are assembling a sequence yourself and do not want a second BOS in the middle of it — a genuinely common bug when people concatenate a chat template with a manual encode.

Truncation is silent

```
enc = tok(long_text, truncation=True, max_length=512)
```

Without `truncation=True`, a too-long input raises or produces a tensor the model rejects. With it, the text is cut and nothing tells you. A classifier handed a 3,000-word complaint reads the first 512 tokens — which, for a complaint, is the polite introduction rather than the accusation at the end.

EXAMINATION

Hugging Face, End to End — final examination

This paper covers the whole course:

- reading a repository and its licence
- tracing text through the transformers library
- fitting a model to the hardware in front of you
- building and debugging a Space
- working with datasets far larger than your RAM
- building and measuring retrieval
- adapting a model with LoRA or QLoRA
- depending on all of it without being surprised later

56 questions, the whole pool. The site draws 25 for a sitting, pass mark 70%. Work any 25 under your own conditions. Answers from page 490.

1 Module 1 · The Hub, read properly

You want to know what a large repository will cost in disk before you fetch anything. Which answer is reliable?

A. Add up every one of the file sizes shown in the Files and versions listing on the page.

B. Multiply the parameter count in the repository′s name by two bytes each.

C. Read total_size in the safetensors index, or call model_info with files_metadata.

D. Check the download counter, which scales with how large the weights are.

2 Module 1 · The Hub, read properly

A repository ships both .bin and .safetensors copies of the same weights. What does a naive snapshot_download do?

A. Fetches both, doubling the download; allow_patterns takes only what you need.

B. Fetches the safetensors copy only, because that format is always preferred by default.

C. Fetches whichever copy is newer according to the repository′s commit history.

D. Fails with an error, because two formats of one model cannot both be cached.

3 Module 1 · The Hub, read properly

Why can Hub search fail to return the most-used model for a task even when that model plainly exists?

A. Search only indexes repositories updated within the last twelve months.

B. Search matches repository names and card text rather than what a model does.

C. Search ranks by likes, and heavily used models attract relatively few likes.

D. Search excludes any repository whose card omits a pipeline_tag in front matter.

4 Module 1 · The Hub, read properly

You click Agree and access repository on a gated model. What has that settled?

A. That the model is licensed permissively, because access was granted at all.

B. That your organisation's accounts now have access to the same repository too.

C. That you may download the files; the licence still governs what you do next.

D. That the terms have been accepted for every account you use on the platform.

5 Module 1 · The Hub, read properly

Which facet of the dataset filters is computed automatically rather than claimed by the uploader?

A. The language tags, which the viewer detects from a sample of rows.

B. The licence, which the Hub verifies against the LICENSE file in the repository itself.

C. The task category, drawn from a controlled vocabulary the Hub maintains.

D. The size band, which is populated from the dataset viewer rather than typed.

Module 1 · The Hub, read properly

The module starts on p. 11.

MODULE 1 TEST

Module 1 test, Q1, p. 56

A transcription script has not been edited since January, yet the transcripts it produced in July differ from those it produced in March. What is…

Answer B: from_pretrained without a revision loads whatever is on the main branch that day.

Why: A repository on the Hub is a git repository, and main moves. Every loading function takes a revision argument that accepts a branch, a tag or a commit hash. Without one, the code means whatever is on main at the moment it runs, so the model can change without the code changing. Pinning is one argument and removes the whole category.

Module 1 test, Q2, p. 56

You clone a model repository with plain git and find that model.safetensors is about 130 bytes. What has happened?

Answer C: Git cloned the pointer; the bytes live in large-file storage and were never fetched.

Why: Git handles source code, not five-gigabyte tensors, so the Hub keeps pointers in git and bytes in a separate large-file store. Without the large-file client installed, git faithfully clones what was committed: a few lines of text naming an object hash. Nothing failed, and hf download or hf_hub_download resolves the real object.

Module 1 test, Q3, p. 57

A model shows four million downloads over thirty days. What is the strongest inference you can draw from that number?

Answer A: Enough different code paths have loaded it that obvious breakages are known.

Why: The count is file resolutions over a rolling thirty days, dominated by continuous integration, library defaults and containers with cold caches. That makes it a floor on how well exercised a model is: the missing tokenizer file and the unparseable config have already been found by somebody. It is not a ranking, because it measures neither quality nor fit.

Module 1 test, Q4, p. 57

Your laptop cannot run a model at full precision. What is the fastest route to a version that fits?

Answer C: Open the base model's page and follow its Quantizations link.

Why: Cards declare their parent with a base_model field and the Hub indexes the relationship in both directions, so a base model's page carries links reading Finetunes, Quantizations and Adapters. Following Quantizations gives you every GGUF, AWQ and GPTQ conversion anyone has published, already filtered. Search matches names and card text, not meaning, so it would miss most of them.

Module 1 test, Q5, p. 57

A card tags a fine-tune apache-2.0. Its base_model field names a repository under a non-commercial research licence. Which terms govern your paid product?

Answer D: The non-commercial terms, because a derivative inherits its base's terms.

Why: A fine-tune is a derivative of the model it was built on, so the base's terms travel with it however the child is tagged. Mis-tagging is common, sometimes careless and sometimes a sincere belief that an adapter is a separate artefact. The check is to follow base_model until you reach a repository with no parent, and read that licence.

Module 1 test, Q6, p. 58

A repository ships only safetensors weights, and its usage snippet passes trust_remote_code=True. What risk remains?

Answer B: Python from the repository is downloaded and imported when the model loads.

Why: Safetensors removes one hole: a JSON header plus raw bytes has no code path. The flag opens a different one, by importing modelling code from the repository into your process. The weights format and the loading code are two separate decisions, so read the .py files, pin the revision beside the flag, and run it where no tokens live.

THE LESSON CHECKS

Lesson 1, p. 17

Someone runs `git clone` on a model repository and ends up with a `model.safetensors` of about 130 bytes containing a few lines of text. What…

Answer C: Git cloned the pointer file that stands in for the weights; the actual bytes live in a large-file store that Git LFS or `hf download` fetches separately.

Why: A model on the Hub is a git repository with big files stored beside it, so it has versions, history and pull requests — and if you do not pin a revision, the model under your code can change without your code changing.

A The repository is gated and they…: Assumes an access control would hand back a stub file rather than refuse the request outright.

B The transfer was cut short by…: Reads a consistent, exactly-sized text file as a partial binary download.

D The branch `main` had moved to…: Blames repository versioning for a checkout that reproduced exactly what was committed.

Lesson 2, p. 21

A team needs a 4-bit version of a model they have already chosen so it fits on a laptop. What is the reliable way to…

Answer B: Open the base model's page and follow its Quantizations link, which indexes every published conversion.

Why: Hub search matches repository names and card text rather than meaning, so the filter sidebar — task tag, library, licence, and the base-model links to fine-tunes and quantizations — is the real interface, and the sort order answers popularity rather than quality.

A Search for the model name plus…: Relies on a naming habit rather than indexed metadata; many conversions use different suffixes, and name search will silently miss them.

C Filter the models list to the…: Finds popular quantized models in general, not conversions of the specific model already chosen, and excludes AWQ and GPTQ formats entirely.

D Use full-text card search for the…: Depends on the uploader writing prose rather than on the structured base_model field the Hub actually indexes.

Lesson 3, p. 25

A team wants to sell a product built on a fine-tuned model. Its card says `license: apache-2.0`, but the `base_model` field points at a model…

Answer B: The non-commercial terms of the base flow down to the derivative, so the Apache-2.0 label on the card does not make it safe to sell.

Why: Check the licence family and the base model before anything else, because a fine-tune inherits its parent's terms no matter what its own card claims.

A The fine-tune is a new work…: Treats training as an act that resets the terms attached to what was trained on.

C Because the repository is not gated…: Confuses the access gate with the licence, when the gate only controls who gets the files.

D The conflict makes the licensing void…: Assumes an unclear licence resolves in the user's favour rather than removing permission.

MAKE THINGS

Hugging Face, End to End

Read a model card licence-first, run a Space for free, keep a token safe, publish something of your own.

WHAT IT TEACHES	The Hub is a host for git repositories that happen to contain neural networks, and that framing explains the rest.
WHAT IS INSIDE	Eight modules and 75 lessons, 28 figures drawn from data, eight case studies, eight module tests, a 56-question examination, and every answer with the reason behind it.
LICENCE	CC BY-NC-SA 4.0. Copy, print, share and adapt it for any non-commercial purpose, with credit, under the same licence. There is no paid edition.
THE COURSE	Free to read, with the lessons, the films and the examination, at **addaly.com/learn/hugging-face**

Scan to open the course

Addaly

First edition, September 2026 · build 62a26075