THE AI TOOLCHAIN

NO. 011

Head

THE WEEK'S TOP PICKS

PUBLISHED OCTOBER 30, 2026 · FRIDAYS

// SHARE THIS WEEK ▽ Top 5 ▽ Top Pick

The best tooling updates that shipped this week.

The week in numbers · Oct 23–29

1,670
changes we read →

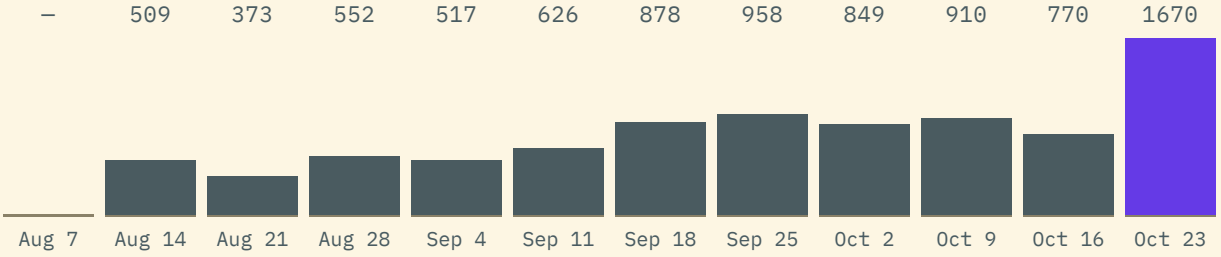
416
tools shipped →

2
product launches →

912
breaking changes →

Changes per week, last 12 weeks · Oct 23–29

Changes we read in that week's dailies, including docs and API changes. A gap means no daily was published.



▸ // HOW THESE PICKS ARE MADE SHOW

▾ // BENCHMARKS MOVED FOR TOOLS WE TRACK HIDE

DeepSeek Models APEX-Agents DeepSeek V4.1 Flash 1017 max

vim ? help

69.8% +30.3 pts released 2026-10-17 · leaderboard ↗

Mistral AI APEX-Agents Mistral Large 4 high

51.6% first result released 2026-10-06 · leaderboard ↗

OpenAI APEX-Agents GPT-6.1 Terra max

68.3% +25.3 pts released 2026-10-26 · leaderboard ↗

DeepSeek Models CursorBench DeepSeek V4.1 Flash 1017 high

45.6% first result released 2026-10-17 · leaderboard ↗

DeepSeek Models CursorBench DeepSeek V4.1 Flash 1017 low

44.9% first result released 2026-10-17 · leaderboard ↗

DeepSeek Models CursorBench DeepSeek V4.1 Flash 1017 medium

46.3% first result released 2026-10-17 · leaderboard ↗

DeepSeek Models CursorBench DeepSeek V4.1 Flash 1017 max

47.7% first result released 2026-10-17 · leaderboard ↗

DeepSeek Models ARC-AGI-2 DeepSeek V4.1 Flash 1017 max

78.1% +5.2 pts released 2026-10-17 · leaderboard ↗

DeepSeek Models ARC-AGI-2 DeepSeek V4.1 Flash 1017 low

73.6% +0.7 pts released 2026-10-17 · leaderboard ↗

Anthropic ARC-AGI-2 Claude Sonnet 5.5 low

24.0% −37.3 pts released 2026-09-28 · leaderboard ↗

Anthropic ARC-AGI-2 Claude Sonnet 5.5 high

67.4% +6.1 pts released 2026-09-28 · leaderboard ↗

Anthropic ARC-AGI-2 Claude Sonnet 5.5 medium

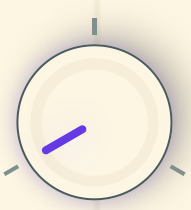
32.8% −28.5 pts released 2026-09-28 · leaderboard ↗

Scores from Epoch AI Benchmarking Hub (CC BY 4.0), evaluated 2026-10-23 to 2026-10-30.

VIEW

ISSUE VIEW

scroll



SCROLL | EXPAND IN PLACE | PREVIEW PANE

INFORMATION DENSITY

everything



BRIEFED | **EVERYTHING**

Every change in this week's picks, in full.

all 47 changes

Do you prefer **scroll**? 👍 👎

This week's 5

01	**PyTorch**	Feature · Model Training & Fine-Tuning
02	**Hugging Face Transformers**	Feature · Agentic Infrastructure
03	**Cursor**	Feature · AI Coding Agents
04	**Replit Agent**	Feature · AI Coding Agents
05	**Pipecat**	Feature · AI Agent Frameworks

The picks in full

01 PyTorch v2.15.0

Model Training & Fine-Tuning

Tensors and Dynamic neural networks in Python with strong GPU acceleration

// WHY IT MADE THE LIST

DEPTH 5/5 IMPACT 4/5 NOVELTY 4/5

PyTorch 2.15 adds FP8, `_scaled_mm`, and a `grouped_mm` with a decode fast path on Apple Silicon, a rewritten NCCL backend with a hang-dump flight recorder, one-sided NIXL transports, and ROCm 10.0 pip wheels. The Apple Silicon FP8/MoE path and installable ROCm open training and serving work the default wheel could not do before.

> PyTorch 2.15 replaces the default NCCL backend with a rewritten implementation (flight recorder, legacy/lazy fallbacks), adds FP8 and new streaming APIs on Apple Silicon, ROCm 10.0 wheels for AMD gfx950/gfx1250, and CUDA 13.2/13.4 wheels with experimental Rubin support, alongside FSDP2 gradient-precision controls and a long list of breaking changes to serialization, checkpointing and exception handling.

└──▷ **WHAT SHIPPED · 15 FEATURES** most completely described first ? what's the number?

01 Distributed P2P communication backend changes

BREAKING

95

`backend='nccl'` now creates a rewritten NCCL implementation with a hang-dump flight recorder, with rollback via `TORCH_DIST_USE_NCCL2=0` or `backend='nccl-legacy'` (available only in this release, removed next) or opt-in to independent per-peer P2P communicators via `backend='nccl-lazy'`; separately, per-directed-edge pipeline P2P communicators can now be enabled with `dist_config.pipeline_per_edge_p2p = True` or `TORCH_DISTRIBUTED_PIPELINE_PER_EDGE_P2P=1`, renaming `torch.distributed.config.pipeline_per_direction_p2p` to `pipeline_per_edge_p2p` (the old attribute now raises `AttributeError` and `TORCH_DISTRIBUTED_PIPELINE_PER_DIRECTION_P2P` is silently ignored).

Initialize a process group with the new NCCL backend and opt into lazy independent P2P communicators for overlapping concurrent sends to different peers.

```python
import torch
import torch.distributed as dist

# Eager communicator creation, one device per group
dist.init_process_group('nccl', device_id=torch.device('cuda',
local_rank))

# Or: independent P2P communicators per peer pair
dist.init_process_group('nccl-lazy',
device_id=torch.device('cuda', local_rank))
```

Keep a distributed training job on the old NCCL backend while validating the new default in staging.

```
$ TORCH_DIST_USE_NCCL2=0 torchrun --nproc_per_node=8 train.py
```

— Names exact backend flags and rollback env var; includes runnable example. Names config key, env var and the renamed attribute exactly.

v2.15.0 launch-20261029-bfb5cbc8

---

02 **FSDP2 gradient-precision controls** BREAKING 95

FSDP2 now supports fp32 gradient accumulation and reduction for precision-sensitive parameters via `MixedPrecisionPolicy`'s `param_dtype_override_fn` (e.g. keeping gate/norm params in fp32), CUDA-graph-captured gradient accumulation, and gradient reduction overlapped with compute. With `reduce_dtype=None`, gradients now accumulate directly in `.grad` following each parameter's `grad_dtype`, changing the previous default where gradients accumulated and reduced in `param_dtype`; set `reduce_dtype` explicitly to restore the 2.14 behaviour.

Keep MoE router gates and norm weights in fp32 while training the rest of the model in bf16 with FSDP2.

```python
from torch.distributed.fsdp import MixedPrecisionPolicy,
fully_shard
import torch

def keep_fp32(param_name, param):
    return 'gate' in param_name or 'norm' in param_name

policy = MixedPrecisionPolicy(
    param_dtype=torch.bfloat16,
    param_dtype_override_fn=keep_fp32,
)
fully_shard(model, mp_policy=policy)
```

Keep fp32 precision for FSDP2 gradient reduction when training with bf16 compute by setting `reduce_dtype` explicitly on `MixedPrecisionPolicy`.

```python
from torch.distributed.fsdp import fully_shard,
MixedPrecisionPolicy
import torch

# Gradients accumulate and reduce-scatter in fp32 (new 2.15
default)
mp_policy = MixedPrecisionPolicy(param_dtype=torch.bfloat16)
fully_shard(model, mp_policy=mp_policy)

# To keep bf16 reduction (2.14 behaviour), set reduce_dtype
explicitly
mp_policy_bf16 = MixedPrecisionPolicy(param_dtype=torch.bfloat16,
reduce_dtype=torch.bfloat16)
fully_shard(model, mp_policy=mp_policy_bf16)
```

— Runnable code examples show exact API and migration path. v2.15.0

launch-20261029-bfb5cbc8

03 Stricter weights_only default excludes nested tensors

90

BREAKING

Nested jagged and strided nested tensors are removed from the default `weights_only=True` allowlist; loading checkpoints containing them now raises Unsupported global unless you use torch.serialization.safe_globals([_rebuild_njt, NestedTensor]) or `weights_only=False`.

Load a checkpoint that contains nested tensors under the new strict `weights_only=True` default by explicitly allowlisting the required globals.

```python
import torch
from torch.nested._internal.nested_tensor import _rebuild_njt,
NestedTensor

with torch.serialization.safe_globals([_rebuild_njt,
NestedTensor]):
    loaded = torch.load('njt.pt', weights_only=True)
```

— Exact error, API, and runnable migration code given. v2.15.0

04 FP8 and new stream APIs on Apple Silicon NEW

85

Apple Silicon now supports `float8_e4m3fn`, `_scaled_mm`, a new `grouped_mm` with a decode fast path (1.32–1.34x tokens/s on MoE decode), and public `torch.mps.Stream` APIs.

Run FP8 matrix multiplication on Apple Silicon using the new MPS float8 support.

```python
import torch

x = torch.randn(128, 256, device='mps').to(torch.float8_e4m3fn)
w = torch.randn(512, 256, device='mps').to(torch.float8_e4m3fn)
scale_x = torch.tensor(1.0, device='mps')
scale_w = torch.tensor(1.0, device='mps')
out = torch._scaled_mm(x, w.t(), scale_a=scale_x,
scale_b=scale_w)
```

— Runnable example plus exact dtype and API names. v2.15.0 launch-20261029-bfb5cbc8

05 ROCm 10.0 wheels and AMD gfx950/gfx1250 kernel support NEW

75

PyTorch 2.15 adds ROCm 10.0 wheels built on AMD's TheRock packages, with ROCm installed as a pip dependency (the matching `rocm` package) rather than bundled libraries. Inductor now autotunes FlyDSL GEMM and `grouped_mm` templates on AMD gfx950 (requires FlyDSL 0.3.x), and uses the Tensor Data Mover on AMD gfx1250 (MI450) for dense `mm` / `addmm`, pointwise and reduction codegen, and FlexAttention forward and decode; gfx1250 is added to the nightly wheel architecture list.

— Names exact GPU targets and package requirements, no example. v2.15.0 launch-20261029-bfb5cbc8

06 **Selective activation checkpointing default behaviour change**

75

BREAKING

Selective activation checkpointing now saves (rather than recomputes) outputs of operators with an ordered effect when the policy returns `CheckpointPolicy.PREFER_RECOMPUTE`; use `CheckpointPolicy.MUST_RECOMPUTE` to keep the 2.14 behaviour.

— Exact policy enum values given to restore old behavior. v2.15.0

07 **Distributed checkpoint metadata now includes tensor strides**

60

BREAKING

Distributed checkpoint `.metadata` files now include tensor strides (`TensorProperties.strides`), so checkpoints saved with 2.15 cannot be loaded by PyTorch 2.14 or earlier (CheckpointException: too many values to unpack (expected 5)).

— Names exact field and error, no migration path given. v2.15.0

THINNER COVERAGE BELOW

08 **CUDA 13.2/13.4 wheels with experimental Rubin support**

IMPROVED

55

The default PyTorch wheel now uses CUDA 13.2 (13.2.2 toolkit), with CUDA 13.4 GA wheels available alongside it for Linux x86_64, Linux aarch64, and Windows, including experimental support for NVIDIA Rubin GPUs.

— Version numbers given but no install command shown. launch-20261029-bfb5cbc8 v2.15.0

09 **torch.ops exception and warning translation change**

BREAKING

50

Operators called through `torch.ops.*` and `torch._C._dispatch_call_boxed` now use PyTorch's own exception and warning translation, so user-defined pybind11 exception translation is lost and `TORCH_WARN` warnings now reach Python's `warnings` module.

— Names affected APIs but no migration example given. v2.15.0

10 **Python 3.11 through 3.15 support, including free-threaded builds**

NEW

45

PyTorch now supports Python 3.11 through 3.15, including free-threaded `3.14t` and `3.15t` builds, with Dynamo adding `torch.compile` support for Python 3.15.

— States supported versions, no usage example. launch-20261029-bfb5cbc8 v2.15.0

11 **Pipeline send budget and staggered FSDP unshards** IMPROVED

45

Pipeline parallelism now accepts a `max_outstanding_sends` budget and staggers FSDP unshards across each rank's startup bubble instead of issuing all at the start of a step.

— Names the new budget parameter but no usage example. launch-20261029-bfb5cbc8

12 **importlib.reload on torch now raises ImportError** BREAKING

40

importlib.reload(torch) and re-importing `torch` after a failed import now raise `ImportError` instead of producing misleading errors or segfaults.

— States behavior change, no example or workaround shown. v2.15.0

13 **One-sided NIXL transport backend** NEW

35

PyTorch now adds one-sided transports for `torch.distributed` with a NIXL backend supporting CUDA-graph-capturable transfers.

— No API name or usage shown, just capability stated. v2.15.0

14 **Faster NVFP4 decoding on Blackwell GPUs** IMPROVED

35

NVIDIA Blackwell GPU LLM decoding at 4-bit (NVFP4) precision is now faster through fewer, better-overlapped NVGEMM kernel launches.

— No API, flag or example given for enabling this. v2.15.0

15 **Intel GPU SYCL kernel authoring and profiling** IMPROVED

20

Intel GPU support gains simpler SYCL kernel authoring and multi-GPU profiling.

— Vague description with no named API or mechanism. v2.15.0

└──▷ BREAKING ON UPGRADE

! FSDP2 `reduce_dtype=None` now causes gradients to accumulate directly in `.grad` following each parameter's `grad_dtype`, changing the previous default behavior.

! importlib.reload(torch) and re-importing `torch` after a failed import now raise `ImportError` instead of producing misleading errors or segfaults.

! Nested jagged and strided nested tensors are removed from the default `weights_only=True` allowlist; loading checkpoints containing them now raises `Unsupported global` unless you use `torch.serialization.safe_globals` or `weights_only=False`.

! Operators called through `torch.ops.*` and `torch._C._dispatch_call_boxed` now use PyTorch's own exception and warning translation, so user-defined pybind11 exception translation is lost and `TORCH_WARN` warnings now reach Python's `warnings` module.

! Selective activation checkpointing now saves (rather than recomputes) outputs of operators with an ordered effect when the policy returns `CheckpointPolicy.PREFER_RECOMPUTE`; use `CheckpointPolicy.MUST_RECOMPUTE` to keep the 2.14 behaviour.

! `backend='nccl'` now creates the rewritten NCCL implementation; the legacy backend is available as `backend='nccl-legacy'` or `TORCH_DIST_USE_NCCL2=0` for this release only — both will be removed in the next release.

! Distributed checkpoint `.metadata` files now include tensor strides (`TensorProperties.strides`), so checkpoints saved with 2.15 cannot be loaded by PyTorch 2.14 or earlier (CheckpointException: too many values to unpack (expected 5)).

! With MixedPrecisionPolicy(reduce_dtype=None), `fully_shard` now accumulates and reduces gradients in the parameter's original dtype instead of `param_dtype`; set `reduce_dtype` explicitly to restore 2.14 behaviour.

! `torch.distributed.config.pipeline_per_direction_p2p` is renamed to `pipeline_per_edge_p2p`; the old attribute raises `AttributeError` and `TORCH_DISTRIBUTED_PIPELINE_PER_DIRECTION_P2P` is silently ignored.

GOOD PICK? 👍 👎 GOOD REASON? 👍 👎

02 Hugging Face Transformers v5.22.0

Agentic Infrastructure

Transformers: the model-definition framework for state-of-the-art machine learning models in text, vision, audio, and multimodal models, for both inference and training.

// WHY IT MADE THE LIST — DEPTH 5/5 — IMPACT 5/5 — NOVELTY 4/5

Train LoRA and PEFT adapters directly on FP8-quantized weights, with the MoE backward pass cutting peak memory from ~28.9 GiB to ~2.8 GiB at 8k tokens — making adapter fine-tuning of huge quantized models feasible on modest hardware; the same release adds Mistral Large 4, a 1T-parameter multimodal MoE with a 1M-token context window.

> Transformers v5.21 adds six new models spanning OCR, TTS, and mixture-of-experts architectures, alongside an FP8 KV-cache backend and FP4 expert quantization for memory-efficient inference, plus breaking changes to tensor-parallel and MoE sharding plans.

└──▷ WHAT SHIPPED · 6 FEATURES most completely described first ? what's the number?

01 FP8 KV-cache backend for generation NEW 90

Adds `cache_implementation="quantized"` with `cache_config={"backend": "fp8"}` to `QuantizedCache`, storing KV states in `torch.float8_e4m3fn` with a single scale per tensor, alongside the existing `quanto` and `hqq` backends.

Reduce KV-cache memory footprint during inference by storing keys and values in FP8 rather than FP16/BF16.

python

```python
from transformers import AutoModelForCausalLM, GenerationConfig

model = AutoModelForCausalLM.from_pretrained("meta-llama/Meta-Llama-3-8B-Instruct")

output = model.generate(
    input_ids,
    cache_implementation="quantized",
    cache_config={"backend": "fp8"},
    max_new_tokens=512,
)
```

— Named config keys plus a runnable generate() example v5.21

02 **FP4 expert quantization for FineGrainedFP8Config** NEW

80

Adds `expert_dtype` option to `FineGrainedFP8Config` (accepts `"fp8"` (default) or `"fp4"`) so DeepSeek-V4.1 FP4 experts can be quantized correctly while other layers stay FP8.

Run DeepSeek-V4.1 with FP4 experts to cut expert-weight memory while keeping other layers in FP8.

python

```python
from transformers import AutoModelForCausalLM,
FineGrainedFP8Config

quant_config = FineGrainedFP8Config(expert_dtype="fp4")
model = AutoModelForCausalLM.from_pretrained(
    "deepseek-ai/DeepSeek-V4.1",
    quantization_config=quant_config,
    device_map="auto",
)
```

— Named config field with default and a runnable example v5.21

03 **New model additions** NEW

75

Adds support for five new models: `UltraVBert`, a visual document understanding and retrieval model combining an `UltraBERT` text encoder with a SigLIP vision encoder, with a `max_image_tiles` setting (default 27) capping 512-pixel tiles per page; `MiniCPM-OCR`, pairing a SigLIP vision encoder with a Qwen3.5 LM for text-dense images such as scanned pages, screenshots, receipts, and tables; Flint, Recursive's mixture-of-experts LM with hybrid sliding-window and full attention, learned attention sinks, sigmoid-routed sparse experts (6 of 256), and a shared dense expert; Gradium Phonon TTS, an on-device TTS model emitting 32-dimensional audio latents at 12.5 Hz decoded to 24 kHz audio via a one-step flow head; and Pocket TTS, Kyutai's 100M-parameter CPU-capable TTS model with voice cloning from recordings, precomputed latents, or cached KV states, plus `PocketTTSAudioTokenizer`, a continuous 32-dimensional VAE-GAN codec usable as the `audio_tokenizer` of `PocketTTSProcessor` or standalone.

— Architecture and a named default config value, no usage example Clear use case but no config or code surface given Detailed architecture with specific expert-routing numbers Specific latent dimensions and sample rates, no code example Named classes and parameters give a clear integration point

v5.21

04 **Breaking changes to tensor-parallel and MoE sharding plans**

70

BREAKING

Tensor parallelism now uses a dedicated embedding style and shards input embeddings by default; `ColwiseParallel` and `RowwiseParallel` apply only to linear layers, so custom TP plans that applied them to embeddings must switch to the new embedding style. Every MoE TP and EP plan now shards its experts across ranks, so custom or downstream MoE parallel plans must add expert sharding or they will fail the new `test_moe_parallel_plans_shard_experts` test and waste memory keeping all experts on every rank.

— Explains before/after behaviour and names the exact test and classes affected v5.21

THINNER COVERAGE BELOW

05 **Linear-attention cache state accessors** IMPROVED 45

Adds new `is_recording_past`, `get_recurrent_state`, and conv-state getters to linear-attention caches, replacing direct internal-state access.

— Named methods but no usage context given v5.21

06 **Tensor-parallel plan for Nemotron** NEW 25

Adds a tensor-parallel plan for Nemotron models.

— Bare one-line mention with no mechanism or config v5.21

NEW INTEGRATION Mistral Large 4 model support

NEW INTEGRATION Nemotron 3.5 Transcribe model support

└──▷ BREAKING ON UPGRADE

! Tensor parallelism now uses a dedicated embedding style and shards input embeddings by default; `ColwiseParallel` and `RowwiseParallel` apply only to linear layers — custom TP plans that applied them to embeddings must switch to the new embedding style.

! Every MoE TP and EP plan now shards its experts across ranks; custom or downstream MoE parallel plans must add expert sharding or models will fail the new `test_moe_parallel_plans_shard_experts` test and waste memory keeping all experts on every rank.

v5.22.0

Transformers v5.22.0 enables adapter training on FP8-quantized weights with chunked MoE backward cutting peak memory from ~28.9 GiB to ~2.8 GiB at 8k tokens.

└──▷ GET THIS VERSION ▾

```
$ git clone --branch v5.22.0
https://github.com/huggingface/transformers.git
# already have the repo? check out this version:
$ git checkout v5.22.0
```

Train LoRA and other PEFT adapters directly on top of FP8-quantized weights; `is_trainable` is now `True` for FP8 layers, and the MoE layer backward pass now dequantizes experts in chunks, cutting peak memory from ~28.9 GiB to ~2.8 GiB at 8k tokens.

NEW INTEGRATION Mistral Large 4 model support

NEW INTEGRATION Nemotron 3.5 Transcribe model support

└─▷ **BREAKING ON UPGRADE**

! Expert-parallel (EP) logic has been separated into its own file (#49679); any code importing EP symbols from their previous location will break.

GOOD PICK? 👍 👎 GOOD REASON? 👍 👎

03 Cursor

announced · Oct 20, 2026

AI Coding Agents

Cursor is an AI-powered code editor built on VS Code that uses large language models to assist with writing, debugging, and refactoring code.

// WHY IT MADE THE LIST

DEPTH 4/5 IMPACT 4/5 NOVELTY 4/5

The new Agent SDK lets you define custom agents in TypeScript, iterate locally in a live playground, and deploy them to Cursor-managed infrastructure triggered by GitHub, GitLab, or Slack events — a coding-IDE vendor's first framework for building and hosting agents.

> Cursor launched the Agent SDK, a framework for defining, testing, and deploying custom agents to Cursor's own infrastructure or a self-hosted runtime.

└──▷ WHAT SHIPPED · 6 FEATURES most completely described first ? what's the number?

01 Define and configure agents with the Agent SDK NEW

95

Agents are defined in TypeScript using `defineAgent` from `@cursor/agent-sdk`, with prompts written in Markdown and tools implemented as TypeScript functions. The agent model is configured via the `model` key, including `id` and `params` entries such as `effort` and `fast`, alongside a `local.cwd` working directory setting.

Define and configure a new agent locally before deploying — use `defineAgent` to set the model, effort, and working directory.

typescript

```typescript
import { homedir } from 'node:os';
import { join } from 'node:path';
import { defineAgent } from '@cursor/agent-sdk';

export default defineAgent({
  name: 'Robo',
  model: {
    id: 'composer-2.5',
    params: [
      { id: 'effort', value: 'high' },
      { id: 'fast', value: 'true' },
    ],
  },
  local: {
    cwd: join(homedir(), '.cache', 'factory', 'robo'),
  },
});
```

— Names exact API, config keys and shows a working code example. launch-20261020-3303f22e

02 **Deploy agents to Cursor infrastructure** NEW 75

The `agent-sdk deploy` command builds the agent, gives it a URL, and starts it on Cursor infrastructure.

After iterating locally and validating with evals, deploy the agent to Cursor infrastructure to give it a live URL.

```
$ agent-sdk deploy
```

— Exact runnable command with clear outcome described. launch-20261020-3303f22e

THINNER COVERAGE BELOW

03 **PR-triggered agent automation** NEW INTEGRATION NEW 55

Agents can be triggered from pull request events on GitHub, GitLab, or Bitbucket to automate code review, risk assessment, and routing workflows.

— Names platforms and use cases but no concrete config shown. launch-20261020-3303f22e

04 **Built-in evals and event replay** NEW 45

Agents support running evals and event replay out of the box, allowing iterative development and workflow validation before deploying.

— Describes purpose but no commands or config named. launch-20261020-3303f22e

05 **Multi-agent patterns** NEW 45

The SDK supports multi-agent patterns: handing tasks to specialized subagents with their own tools and sessions, or dividing work among subagents running in parallel.

— Explains mechanism in prose without named APIs or examples. launch-20261020-3303f22e

06 **Self-hosted agent runtime** NEW 45

Agents can run on your own infrastructure as a single Node process, which can be packaged in a container, independent of Cursor-hosted deployment.

— Names runtime shape but gives no setup steps or commands. launch-20261020-3303f22e

NEW INTEGRATION Artemis Security integration

> updated · Oct 27, 2026
>
> Cursor launches the Agent SDK — a CLI and folder-based framework for defining, testing locally, and deploying code-driven bots to Cursor-managed infrastructure.
>
> Introduces the `@cursor/agent-sdk` CLI: bootstrap a new agent project with `npx @cursor/agent-sdk init ./my-bot`, then iterate locally using `agent-sdk dev` to spin up a bot with a live playground.
>
> Adds `--template` flag to `agent-sdk init` to scaffold from built-in starting points: security reviewer, owners-policy approver, PR autofixer, Linear triager, or Jira triager.
>
> Adds `agent-sdk deploy` to publish an agent to Cursor-managed infrastructure, giving it a URL and access to dashboard-managed secrets and integrations.
>
> Adds `agent-sdk validate` to lint an agent project and `agent-sdk info` to print the discovered project structure.
>
> Adds `agent-sdk eval` for assertion-based testing of bot behavior — checks which tools ran, with what input, and the shape of the reply — runnable locally or in CI.
>
> +2 more
>
> NEW INTEGRATION Artemis Security integration

GOOD PICK? 👍 👎 GOOD REASON? 👍 👎

04 Replit Agent

announced · Oct 27, 2026

AI Coding Agents

Replit Agent is an AI assistant that helps developers code by understanding natural language requests and generating or modifying code within the Replit IDE.

// WHY IT MADE THE LIST

DEPTH 5/5 IMPACT 5/5 NOVELTY 4/5

Replit's Cloud Agents connect existing company GitHub/GitLab repos so non-engineers can describe a change in plain language and get a tested pull request — validated in isolated cloud sandboxes with clickthrough screenshots attached — while a new Monitoring Agent autonomously investigates, fixes, and deploys production issues. Both close human-in-the-loop gaps that previously demanded an engineer at every step.

> Replit Agent expanded Cloud Agents so anyone across a company — not just engineers — can connect a repo, Slack thread, or mobile app to produce tested, browser-validated pull requests, while a new self-improving skills system and shared VM setup reduce repeat work; separately, a new Monitoring Agent can autonomously investigate, fix, and deploy production incidents.

└──▷ WHAT SHIPPED · 8 FEATURES most completely described first ? what's the number?

01 Autonomous Monitoring Agent for production incidents NEW

93

Replit Monitoring Agent is an autonomous on-call agent that investigates production alerts, identifies root causes, validates fixes, and can deploy them without manual intervention, using access to production logs and metrics to investigate errors, slowdowns, and failed deployments in context. Enable `auto-fix` in Publishing settings so it automatically investigates and validates a fix when an alert fires and emails a summary of what happened and what changed, and enable `auto-publish` in Publishing settings so it can deploy a validated fix to production automatically, with the creator choosing whether to allow this.

Enable fully autonomous incident response so the Monitoring Agent investigates, fixes, and ships repairs to production while you sleep.

📍 In your app's Publishing settings, enable 'monitoring alerts and auto-fix', then turn on `auto-publish` to allow Monitoring Agent to deploy validated fixes automatically.

— Names exact config toggles and their location in Publishing settings.

launch-20261022-af82b9f1

02 **Start and follow pull requests from Slack** NEW 78

Mention or DM `@Replit` in Slack to start an investigation or open a pull request from a conversation thread, with progress and feedback continuing in the same thread; set up by going to Workspace Settings, linking your Replit account, and choosing a workspace.

When a support lead or designer wants to propose a fix without setting up a local environment, go to Workspace Settings to link Replit to Slack, then DM or mention the bot to open a pull request from the conversation.

> 📍 In Replit, go to Workspace Settings › link your Replit account › choose a workspace. Then in Slack, DM or mention @Replit with a description of the change. Follow progress in the thread; the agent opens a tested pull request in GitHub or GitLab with screenshots and video attached.

— Exact bot mention and setup steps given. launch-20261027-2107699a

03 **Shared environment setup via Cloud Agents** NEW 75

Cloud Agents explore the codebase and configure an initial VM with repos, runtimes, dependencies, and credentials pulled from external secrets managers via a credential broker, so every contributor starts from a running, current app without individual setup.

When onboarding a new team or repository so all contributors share one running environment without individual setup.

> 📍 Visit replit.com/repos, connect your repository, and let Cloud Agents explore the codebase and configure the initial VM. All contributors then start from that shared, automatically updated environment.

— Names credential broker and gives exact setup steps. launch-20261027-2107699a

04 **Self-improving workspace memory (skills)** NEW 70

In early access, when engineers approve review feedback, agents draft or update written instructions that all future Cloud Agent contributions follow, reducing token usage by 51% and cutting a common data task from 16 minutes to 1 minute internally.

— Concrete metrics given but no setup or activation steps. launch-20261027-2107699a

05 **Cloud Agents for existing company repos** NEW 65

Connect existing company repositories at `replit.com/repos` to let non-engineers describe changes in plain language and have Cloud Agents produce tested pull requests in GitHub or GitLab; available to all Pro & Enterprise customers.

— Names exact URL and plan tiers but light on internal mechanism.

launch-20261027-2107699a

THINNER COVERAGE BELOW

06 **Automated browser testing for Cloud Agent PRs** NEW

55

Cloud Agents test each change in isolated cloud sandboxes by opening the running app in a browser, clicking through the change, and attaching screenshots and video to the pull request before it reaches a reviewer.

— Clear mechanism but automatic, with no user-facing controls described.

launch-20261027-2107699a

07 **Plain-language codebase Q&A** NEW

30

Replit lets anyone query the company codebase in plain language to get answers grounded in what is actually built, accessible to support, marketing, and sales roles.

— Describes capability and audience but no access path given. launch-20261027-2107699a

08 **Start pull requests from mobile app** NEW

20

Start a pull request from the Replit mobile app when an idea comes up away from a desk.

— Single bare sentence with no mechanism or steps. launch-20261027-2107699a

GOOD PICK? 👍 👎 GOOD REASON? 👍 👎

05 Pipecat v1.15.0

AI Agent Frameworks

Pipecat is a framework for realtime voice and multimodal agents, orchestrating speech-to-text, model inference, and text-to-speech as a streaming pipeline.

// WHY IT MADE THE LIST — DEPTH 4/5 — IMPACT 5/5 — NOVELTY 5/5

Expressive conversation (alpha) lets TTS perform expression tags the LLM writes — `<<excited>>` sets delivery, `[[laugh]]` inserts a sound, `^^word^^` stresses a word — and a PipecatToneReader reads emotion and sarcasm from each transcribed user turn, giving realtime voice agents a perception-and-expression layer no other framework ships.

> Pipecat v1.14.0 adds new LLM provider integrations (Perplexity Agent API, NVIDIA Nemotron Omni multimodal, Smallest AI), a new Zoom Scribe Live STT service, speaker diarization for NVIDIA transcription, and LLM-level text transforms, alongside several breaking default model changes across TTS and LLM services.

└──▷ WHAT SHIPPED · 12 FEATURES most completely described first ? what's the number?

01 Default model and behaviour changes across services

UPDATED INTEGRATION — BREAKING — 95

Several services change default behaviour and require migration: `RimeV1TTSService` now requires an explicit `settings.model`, the previous implicit default no longer applies; `GoogleVertexLLMService` default model changed from `gemini-3.6-flash` to `gemini-3.8-flash` (pass settings=GoogleVertexLLMService.Settings(model='gemini-3.6-flash') to keep the previous model); `NvidiaLLMService` default model changed to `nvidia/nemotron-3.5-super-vl-120b-a12b` (set settings=NvidiaLLMService.Settings(model='nvidia/nemotron-3-super-120b-a12b') to keep the previous model); `SpeechifyHttpTTSService` default model changed to `simba-3.5` (set `settings.model` to `simba-3.2` or `simba-3.0` to keep the previous behaviour); and `strip_markdown` now strips URLs from Markdown links and images by default (register StripMarkdown(strip_urls=False) or pass `strip_urls=False` to `VoiceFormatter` to restore the old behaviour).

— Exact old/new values and migration code for every change v1.14.0

02 Perplexity Agent API LLM service NEW

90

Adds `PerplexityAgentHttpLLMService` for Perplexity's Agent API with web search and function calling. Settings exposes `preset`, `web_search`, and `max_steps`,

defaulting to `openai/gpt-5.6-luna` with web search on.

Use Perplexity's Agent API with web search enabled and a step limit in a voice pipeline.

```python
from pipecat.services.perplexity import
PerplexityAgentHttpLLMService

llm = PerplexityAgentHttpLLMService(
    api_key=os.environ['PERPLEXITY_API_KEY'],
    settings=PerplexityAgentHttpLLMService.Settings(
        model='openai/gpt-5.6-luna',
        web_search=True,
        max_steps=5,
    )
)
```

— Runnable example with exact settings and defaults `v1.14.0`

---

03 **LLM output text transforms** NEW 85

Adds `text_transforms` to `LLMTextProcessor` with add_text_transformer() and remove_text_transformer(), so transforms rewrite LLM output before it splits toward TTS, context, and client. The callable type is `TextTransform` in `pipecat.utils.text.transforms`.

Strip unwanted emoji and stage directions from LLM output before they reach TTS, context, and the client.

```python
from pipecat.utils.text.transforms import TextTransform
import re

def remove_emoji(text: str) -> str:
    return re.sub(r'[\U00010000-\U0010ffff]', '', text,
flags=re.UNICODE)

llm_text_processor.add_text_transformer(('*', remove_emoji))
```

— Runnable example demonstrates transform registration `v1.14.0`

---

04 **NVIDIA Omni multimodal LLM service** NEW 80

Adds `NvidiaOmniLLMService` for NVIDIA Nemotron Omni multimodal models, accepting speech, images, video, and text. The `audio_input` constructor argument (default `True`) can replace the STT stage entirely, and run_multimodal_inference() supports one-off requests outside the conversation.

– Named class and method but no usage example v1.14.0

05 **Twilio API key auth and region support** IMPROVED 75

Adds `api_key_sid` and `api_key_secret` to `TwilioFrameSerializer` so auto hang-up authenticates with a Twilio API key instead of the account Auth Token. The development runner's Twilio serializer now reads `TWILIO_API_KEY`, `TWILIO_API_SECRET`, `TWILIO_REGION`, and `TWILIO_EDGE` environment variables for non-US1 region support.

– Named fields and env vars, no code example v1.14.0

06 **NVIDIA STT speaker diarization** NEW 75

Exposes speaker diarization in `NvidiaSTTService` transcription frames; Settings(speaker_diarization=True, speaker_format='Speaker {speaker}: {text}') labels each contiguous speaker run as its own frame with the NVIDIA speaker tag in `user_id`.

– Settings and format string specified, no full snippet v1.14.0

07 **Zoom Scribe Live STT service** NEW 65

Adds `ZoomAISTTService` for Zoom Scribe Live streaming STT, installable via `pipecat-ai[zoom-ai]`. `Settings.silence_duration_ms` (default 350) controls turn end, and `Settings.diarization` enables speaker labels in `TranscriptionFrame.result`.

– Install and settings named but no code example v1.14.0

THINNER COVERAGE BELOW

08 **Context-hub repos management command** NEW 55

Adds the `pipecat context-hub repos` command via the `pipecat-ai-context-hub` 0.9.0 floor bump, enabling adding and removing indexed repos and letting hub queries wait out a running `refresh`.

– Command named, no full usage example v1.14.0

09 **Speechmatics STT sentence segmentation and sample rate** IMPROVED 50

Adds `emit_sentences` to `SpeechmaticsSTTService.Settings`; when enabled, Speechmatics closes a segment at every sentence boundary so each sentence is emitted as its own final transcription. `SpeechmaticsSTTService` also now supports 8 kHz audio input alongside 16 kHz.

– Setting named but no example of use v1.14.0

10 **RTVI bot-output message type field** NEW 50

Adds `output_type` field to the RTVI `bot-output` message, indicating whether the text is a sentence, word, backchannel, or a custom type such as `'code'`.

— Field named but no usage example v1.14.0

11 **LemonSliceTransport model and sample rate options** IMPROVED 50

`LemonSliceTransport` now accepts a `model` parameter to specify LemonSlice models and an `audio_sample_rate` parameter to set audio input/output sample rate (defaulting to 24 kHz).

— Params named but no example v1.14.0

12 **Smallest AI LLM service** NEW 45

Adds `SmallestLLMService` connecting to Smallest AI's Electron model via the OpenAI-compatible Waves API, offered as `Smallest` in `pipecat init`.

— Thin description, no example or settings detail v1.14.0

NEW INTEGRATION AssemblyAI Streaming TTS integration

NEW INTEGRATION ai-coustics Tyto audio quality scoring integration

UPDATED INTEGRATION AWS Bedrock LLM reasoning output support

UPDATED INTEGRATION Tavus emotion expression from delivery tags

UPDATED INTEGRATION Smallest TTS number pronunciation language setting

└──▷ **BREAKING ON UPGRADE**

! `RimeV1TTSService` now requires an explicit `settings.model`; the previous implicit default no longer applies.

! `GoogleVertexLLMService` default model changed from `gemini-3.6-flash` to `gemini-3.8-flash`; pass `settings=GoogleVertexLLMService.Settings(model='gemini-3.6-flash')` to keep the previous model.

! `NvidiaLLMService` default model changed to `nvidia/nemotron-3.5-super-vl-120b-a12b`; set `settings=NvidiaLLMService.Settings(model='nvidia/nemotron-3-super-120b-a12b')` to keep the previous model.

! `SpeechifyHttpTTSService` default model changed to `simba-3.5`; set `settings.model` to `simba-3.2` or `simba-3.0` to keep the previous behaviour.

! `strip_markdown` now strips URLs from Markdown links and images by default; register StripMarkdown(strip_urls=False) or pass `strip_urls=False` to `VoiceFormatter` to restore the old behaviour.

> v1.15.0
>
> Pipecat adds expressive conversation (alpha), classifier-based turn detection, audio quality scoring, predictive backend delegation, and AssemblyAI TTS.

└──▷ GET THIS VERSION ▾

```
$ git clone --branch v1.15.0 https://github.com/pipecat-ai/pipecat.git
# already have the repo? check out this version:
$ git checkout v1.15.0
```

UPDATED INTEGRATION Enable expressive conversation (alpha) with PipelineParams(enable_expressive_conversation=True): TTS services perform expression tags the LLM writes — `<<excited>>` sets delivery, `[[laugh]]` inserts a sound, `[[pause 500]]` pauses in milliseconds, and `^^word^^` stresses a word — instead of speaking them aloud.

Control which expression tags an LLM offers via PipelineParams(expression_policy=...) with an `ExpressionPolicy`: its `delivery` and `sounds` fields list which tags are offered by universal name, and its `guidance` lines (presets such as `ExpressionGuidance.MATCH_STYLE` and `ExpressionGuidance.SLOW_FOR_DETAILS`, or custom strings) say when to use them.

Add `PipecatToneReader` after the STT service to read each transcribed user turn with the Pipecat Tone model, setting emotions and sarcasm likelihood on `TranscriptionFrame`. Install with `pipecat-ai[tone]`.

Add perception to expressive conversation (alpha): STT services or downstream processors set a `perception` field of `PerceptionSignal`s on `TranscriptionFrame`, `InterimTranscriptionFrame`, and `EagerTranscriptionFrame`; the user aggregator writes the turn's final reading into `UserTurnStoppedMessage.perception`.

Use `ClassifierUserTurnStopStrategy` and `ClassifierUserTurnStrategies` to let a classifier such as `JevClassifier` decide whether a user turn is complete, as an alternative to `FilterIncompleteUserTurnStrategies`.

+3 more

NEW INTEGRATION AssemblyAI Streaming TTS integration

NEW INTEGRATION ai-coustics Tyto audio quality scoring integration

UPDATED INTEGRATION AWS Bedrock LLM reasoning output support

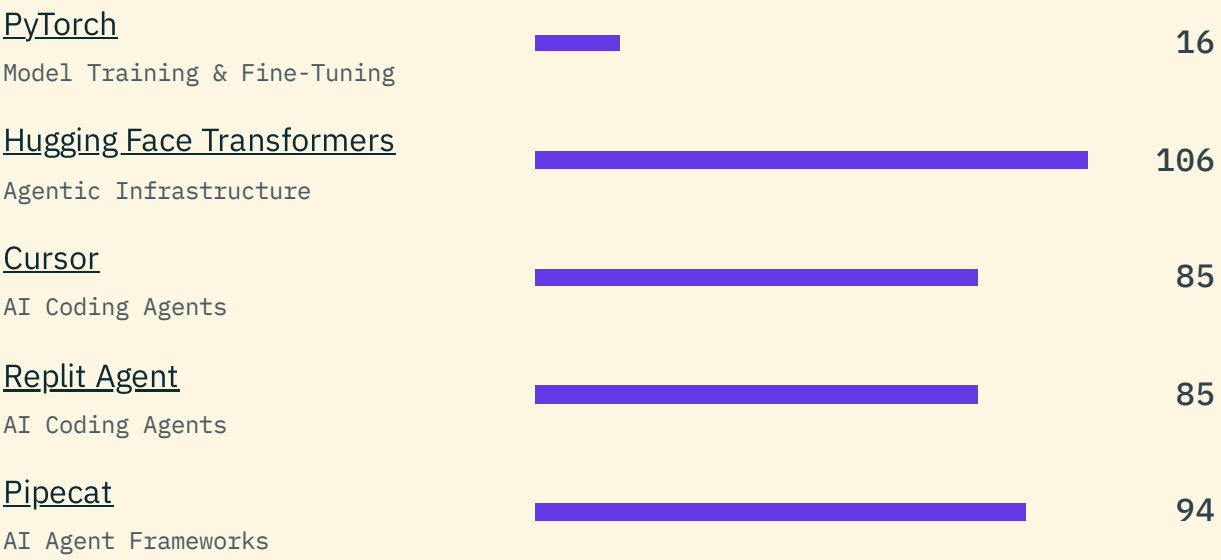
UPDATED INTEGRATION NVIDIA TTS custom model configuration

UPDATED INTEGRATION Smallest TTS number pronunciation language setting

GOOD PICK? 👍 👎 GOOD REASON? 👍 👎

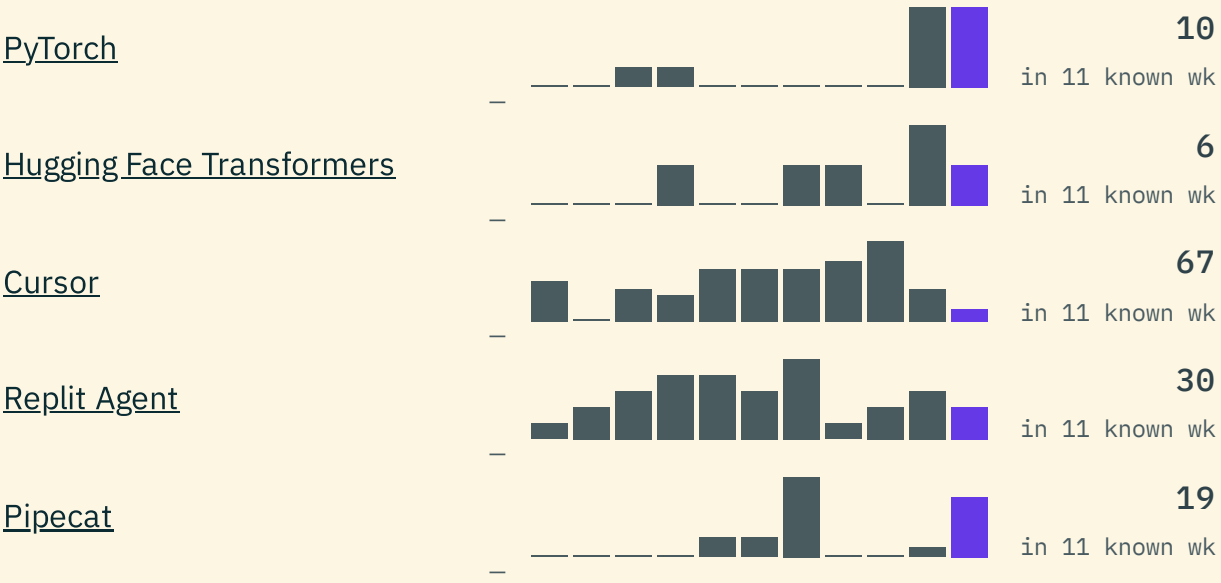
Where each pick stands · Oct 23–29

Tools with changes in each pick's category over the last 12 weeks, including the pick itself.



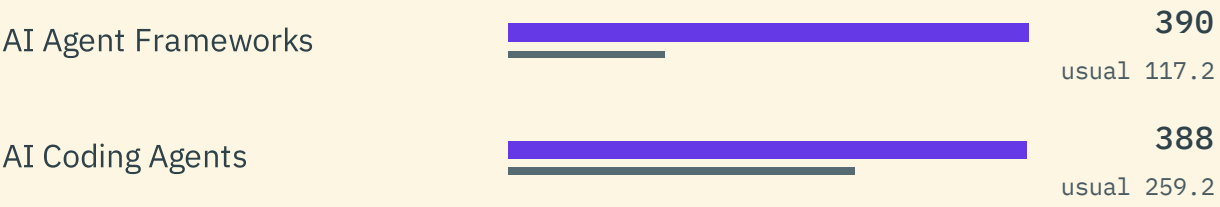
Each pick's release rhythm · Oct 23–29

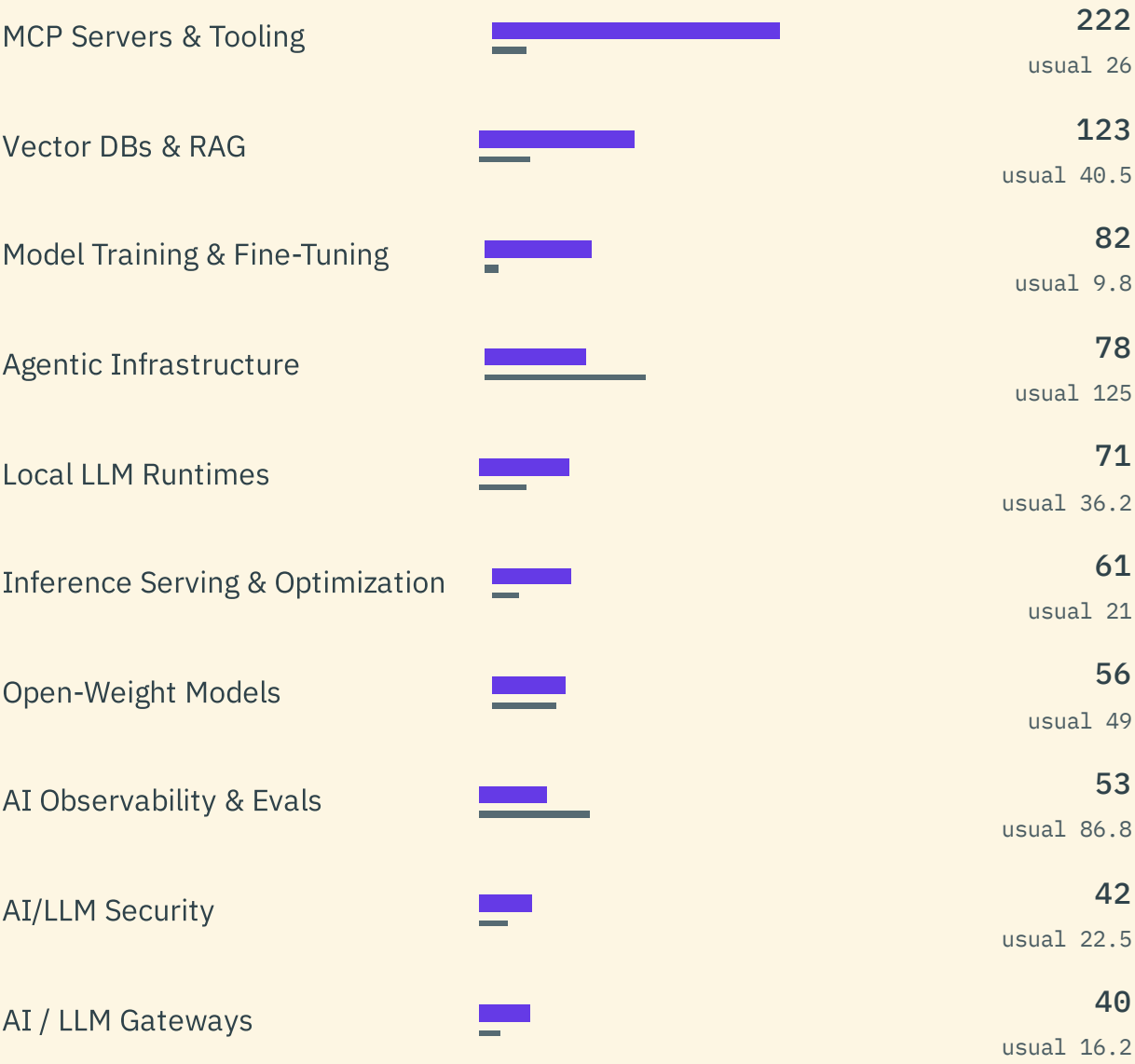
Changes per week for the last 12 weeks. The bright bar is this week; a dash marks a week with no daily.



Which categories moved · Oct 23–29

The thick bar is this week. The thin bar is the average of the four weeks before, counting only weeks with a daily.





Who shipped the most · Oct 23–29

Changes per tool this week. Volume is not a quality score.



// END OF TRANSMISSION

The AI Toolchain · This Week's Highlights · October 30, 2026
The full firehose — every feature release, every day — is in the newsletter.

▷ Subscribe