GitHub Copilot Metrics & Value information

Below is a **comprehensive, API-accurate list of GitHub Copilot metrics** you can retrieve today, structured the way enterprise customers (CTO, Engineering leadership, FinOps) typically consume them. The document now includes:

- A **clear mapping of metrics → CTO business outcomes**
- A **customer-ready KPI table** you can drop into a deck or one-pager

Important context (Feb 2026): Legacy Copilot Metrics APIs are being retired (March–April 2026). All serious reporting should now target the **Copilot Usage Metrics APIs** (public preview, moving to GA).

---

1. API Families (What Exists Today)

A. Legacy Copilot Metrics API (Sunsetting)

Status: Closing April 2, 2026

Scope: Aggregated, IDE-focused metrics (limited depth)

Key characteristics

- Aggregated metrics only
- Max 100 days history
- No PR lifecycle or agent depth
- Replaced by usage metrics APIs

B. Copilot Usage Metrics APIs (Recommended)

Status: Public Preview (Enterprise-ready)

Scope: Enterprise, Organization, User-level (NDJSON)

Key characteristics

- Daily generated reports
- Enterprise, org, and user granularity
- Code generation, chat, agent, PR impact
- Exportable via signed NDJSON links
- Powers the Copilot Usage Dashboard

---

2. Metric Categories & Fields (Copilot Usage Metrics API)

The GitHub Copilot Usage Metrics APIs expose **adoption, productivity, quality/flow, and depth-of-usage signals**. These metrics are designed to answer executive-level questions such as:

- *Are developers actually using Copilot?*
- *Is it saving time and accelerating delivery?*

- *Is it improving flow and throughput?*
- *Are we getting ROI on licenses?*

---

3. Adoption & Engagement Metrics

User Activity

- **Daily Active Users (DAU) –** Unique users who used Copilot on a given day
- **Weekly Active Users (WAU) –** Unique users active over a 7-day window
- **Total Active Users (Monthly) –** Licensed users active in the current calendar month

Adoption

- **Agent Adoption Rate –** % of active Copilot users using agent features
- **IDE Monthly Active Users –** Users interacting with Copilot in IDEs (chat or completions)

4. Code Completion Metrics (Inline Suggestions)

- **Code Completions Suggested –** Total inline suggestions shown
- **Code Completions Accepted –** Suggestions accepted by users
- **Completion Acceptance Rate (%) –** Accepted ÷ suggested

Why this matters: This is the strongest leading indicator of *developer trust and usefulness*.

5. Chat & Agent Usage Metrics

Chat Volume

- **Total Chat Requests**
- **Average Chat Requests per Active User**

Chat Modes

- Ask
- Edit
- Agent

Agent Signals

- **Agent Usage Events**
- **Agent-initiated Code Generation**
- **Agent Adoption %**

---

6. Code Generation Metrics (Lines of Code)

- **Lines of Code Added**

- **Lines of Code Deleted**
- **Net Lines of Code**
- **User-initiated vs Agent-initiated LoC**

Derived from IDE telemetry and surfaced in the Code Generation Dashboard.

7. Language Usage Metrics

- **Language Usage Distribution**
- **Language Usage per Day**
- **Language × Model Breakdown**

8. Model Usage Metrics

- **Model Usage Distribution**
- **Model Usage per Day**
- **Model Usage per Chat Mode**
- **Most Used Chat Model (28-day window)**

---

9. Pull Request Impact Metrics (NEW)

These metrics directly link Copilot to **engineering throughput and flow**:

- **PR Review Suggestions Generated by Copilot**
- **PR Review Suggestions Accepted**
- **Pull Requests Created with Copilot Agent**
- **Pull Requests Created with Copilot Agent that were Merged**
- **Pull Request Cycle Time**
- **Time to Merge (Copilot-influenced)**

This is the first time Copilot → PR throughput correlation is available via API.

---

10. Mapping: Metrics → CTO Business Outcomes

	CTO Outcome	Metrics That Support It	What It Proves
1	**Developer Productivity**	Completion Acceptance Rate, LoC Added, Chat Requests/User	Developers are completing work faster with less manual effort

2	**Faster Time-to-Market**	PR Cycle Time, Time to Merge, Agent-created PRs	Reduced lead time from code to production
3	**Engineering Throughput**	PRs Created & Merged, Agent PR Adoption	More work delivered with the same team size
4	**Developer Experience (DX)**	DAU/WAU, Agent Adoption %, Chat Usage	Engineers trust and rely on Copilot daily
5	**Quality & Flow**	PR Review Suggestions Accepted, Reduced Review Time	Higher-quality PRs and less rework
6	**Cost Efficiency / ROI**	Active Users vs Licenses, Productivity Signals	Value per license justified through measurable output
7	**Platform Alignment**	Language & Model Usage	Copilot aligned to strategic languages and stacks

---

11. Customer-Ready Metrics & KPI Table (Executive View)

This table is designed to be **drop-in ready for customer decks or QBRs**.

	Metric	KPI / Target	Executive Question Answered
1	Monthly Active Users (%)	>70% of licensed users	Are we actually adopting Copilot?
2	Completion Acceptance Rate	25–40%+	Is Copilot useful in daily coding?
3	Agent Adoption Rate	20–30%+	Are teams moving beyond basic autocomplete?
4	Avg Chat Requests / User	Increasing MoM	Are developers using Copilot to unblock work?
5	Net LoC Generated	Trending upward	Is Copilot accelerating delivery?
6	PRs Created with Copilot	Increasing	Is Copilot influencing real delivery artifacts?
7	PR Merge Rate (Agent PRs)	Comparable or higher than baseline	Is Copilot producing production-ready code?
8	PR Cycle Time Reduction	10–30% improvement	Are we shipping faster?
9	Active Users vs Licenses	>85% utilization	Are we getting ROI from licenses?

---

12. What Is Explicitly NOT Included

- Copilot Chat on GitHub.com
- Copilot Mobile
- Copilot CLI
- License counts (use Copilot User Management API)
- Non-IDE telemetry

---

13. Practical Takeaway for Executives

With the Copilot Usage Metrics APIs, organizations can **move the Copilot conversation from "tool usage" to "business outcomes"** by showing:

- Measurable productivity uplift
- Faster delivery and PR flow
- Higher developer satisfaction
- Justified cost per engineer

This enables **CTO-level conversations about scale, ROI, and engineering leverage**, not just feature usage.

---

14. Using These KPIs to Calculate ROI (Customer-Ready Guidance)

This section explains **how to translate Copilot KPIs into a defensible ROI narrative** that resonates with CTOs, CFOs, and FinOps teams. The goal is *not* precision accounting, but **directionally accurate, outcome-based ROI**.

14.1 Core ROI Formula

At a high level, Copilot ROI can be expressed as:

ROI = (Value of Time & Throughput Gained − Cost of Copilot) ÷ Cost of Copilot

Where *value* is derived from productivity, throughput, and flow improvements measured by the KPIs above.

14.2 Step 1 – Establish the Baseline

Before Copilot (or during the first 30 days), capture baseline metrics:

- Average PR cycle time
- Average PRs per developer per month
- Average time spent coding vs waiting/reviewing
- Current release cadence (per team or org)

These baselines allow you to show **before vs after impact**.

14.3 Step 2 – Quantify Productivity Gains

Use Copilot KPIs as *proxies for time saved*:

	KPI	How It Contributes to ROI
1	Completion Acceptance Rate	Indicates reduced manual coding effort
2	Avg Chat Requests / User	Fewer context switches and faster problem solving
3	Net LoC Generated	Faster feature and fix delivery
4	Agent Adoption Rate	Automation of multi-step dev tasks

Example logic:

- If developers save **30–60 minutes per day** (supported by high acceptance and chat usage)
- Multiply by **working days × number of active users**
- Convert time saved into cost using average developer fully loaded cost

---

14.4 Step 3 – Quantify Throughput & Flow Improvements

These KPIs link Copilot directly to delivery speed:

	KPI	ROI Signal
1	PR Cycle Time Reduction	Faster lead time to production
2	PRs Created with Copilot	Increased output per engineer
3	PR Merge Rate (Agent PRs)	Reduced rework and review friction

Business interpretation:

- Faster PRs → faster releases
- Faster releases → earlier value realization
- Earlier value realization → higher business impact per sprint

---

14.5 Step 4 – Translate Improvements into Financial Value

Common customer-friendly translation methods:

1. **Cost Avoidance**
   - Same delivery with fewer engineers added

1. **Productivity Uplift**
    - More features delivered with existing teams
1. **Opportunity Acceleration**
    - Faster delivery of revenue-generating features

Most enterprises position Copilot ROI as **cost avoidance + acceleration**, not headcount reduction.

14.6 Step 5 – Compare Against Copilot Investment

Use:

- **Active Users vs Licenses** → ensure utilization
- **Cost per Developer (Copilot)** → annualized license cost

Then show:

- Cost of Copilot per developer per year
- Estimated value generated per developer per year
- Net value and ROI ratio

14.7 Executive-Ready ROI Summary (Template)

You can summarize ROI to leadership as:

"With GitHub Copilot, **X% of our developers are active monthly**, showing a **Y% code suggestion acceptance rate**. We've reduced **PR cycle time by Z%**, enabling teams to deliver features faster without increasing headcount. Based on time saved and throughput gains, Copilot delivers **multiple-X return on license cost**, positioning it as a scale lever rather than a developer tool."

14.8 Key Guidance for Customer Conversations

- Anchor ROI on **measured KPIs**, not assumptions
- Use **trends and deltas**, not absolute perfection
- Focus on **flow, throughput, and acceleration**
- Tie ROI back to **business priorities** (delivery speed, stability, growth)

This approach keeps the ROI discussion **credible, defensible, and executive-ready**.