Advanced Software-Engineering

DHBW Stuttgart
21.10.2025

Janko Dietzsch

„Aktuelles“ …

- … oder manchmal auch nur etwas lustiges mit tieferem Sinn …
- … etwas das nicht immer zum aktuellen Thema passt – auch nicht nur mit Informatik oder Software zu tun hat ...
- … aber irgendetwas das mir über den Weg läuft und mir interessant und erwähnenswert erscheint

„Aktuelles“ – Stackoverflow Survey 2025

The 2025 Developer Survey is the definitive report on the state of software development. In its fifteenth year, Stack Overflow received over 49,000+ responses from 177 countries across 62 questions focused on 314 different technologies, including new focus on AI agent tools, LLMs and community platforms. This annual Developer Survey provides a crucial snapshot into the needs of the global developer community, focusing on the tools and technologies they use or want to learn more about.

Technology → Admired and Desired

Cargo is the most admired cloud development and infrastructure tool this year

Terraform	15.7%	51.8%
Homebrew	15.2%	56.4%
Cargo	13.9%	70.8%
Make	12.6%	47.6%
APT	11.5%	58.8%

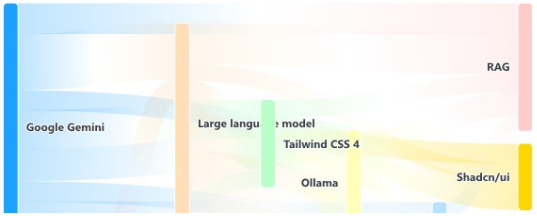
Desired Admired

Rust's growth is directly tied to the success of its build tool and package manager, Cargo, which is the most admired (71%) cloud development and infrastructure tool this year.

Cloud development →

Technology → Worked with vs. want to work with

Developers at all levels are exploring the evolving AI landscape through Stack Overflow



Developers → Profile

USA, Germany and India are top countries responding to this year's survey

United States of America	20.4%
Germany	8.6%
India	7.2%
United Kingdom of Great B	5.8%
France	4%
Canada	3.7%
Ukraine	2.7%
Poland	2.5%
Netherlands	2.5%
Italy	2.4%

Ukraine and France swapped places this year compared to last, placing France in the top 5 list of responding countries.

Country →

AI → AI Agents

A majority of developers don't use AI agents

Yes, I use AI agents at work	14.1%
Yes, I use AI agents at work	9%
Yes, I use AI agents at work	7.8%
No, but I plan to	17.4%
No, I use AI exclusively in c	13.8%
No, and I don't plan to	37.9%

AI agents are not yet mainstream. A majority of developers (52%) either don't use

AI → Sentiment and usage

84% of respondents are using AI tools this year

Yes, I use AI tools daily	47.1%
Yes, I use AI tools weekly	17.7%
Yes, I use AI tools monthly	13.7%
No, but I plan to soon	5.3%
No, and I don't plan to	16.2%

84% of respondents are using or planning to use AI tools in their development process, an increase over last year (76%). This year we can see 51% of professional developers use AI tools daily.

AI tools in the development process →

Technology → Admired and Desired

GitHub is a more desirable collaboration tool than Jira this year

GitHub	59.3%	70.1%
Markdown File	27%	75.8%
GitLab	25.6%	59.5%
Jira	22%	42.1%
Confluence	14.3%	40.2%

Desired Admired

Jira steps down as the most desired tool for code documentation and collaboration and the new top desired tool is GitHub. Markdown continues to be the most admired sync tool for the third year.

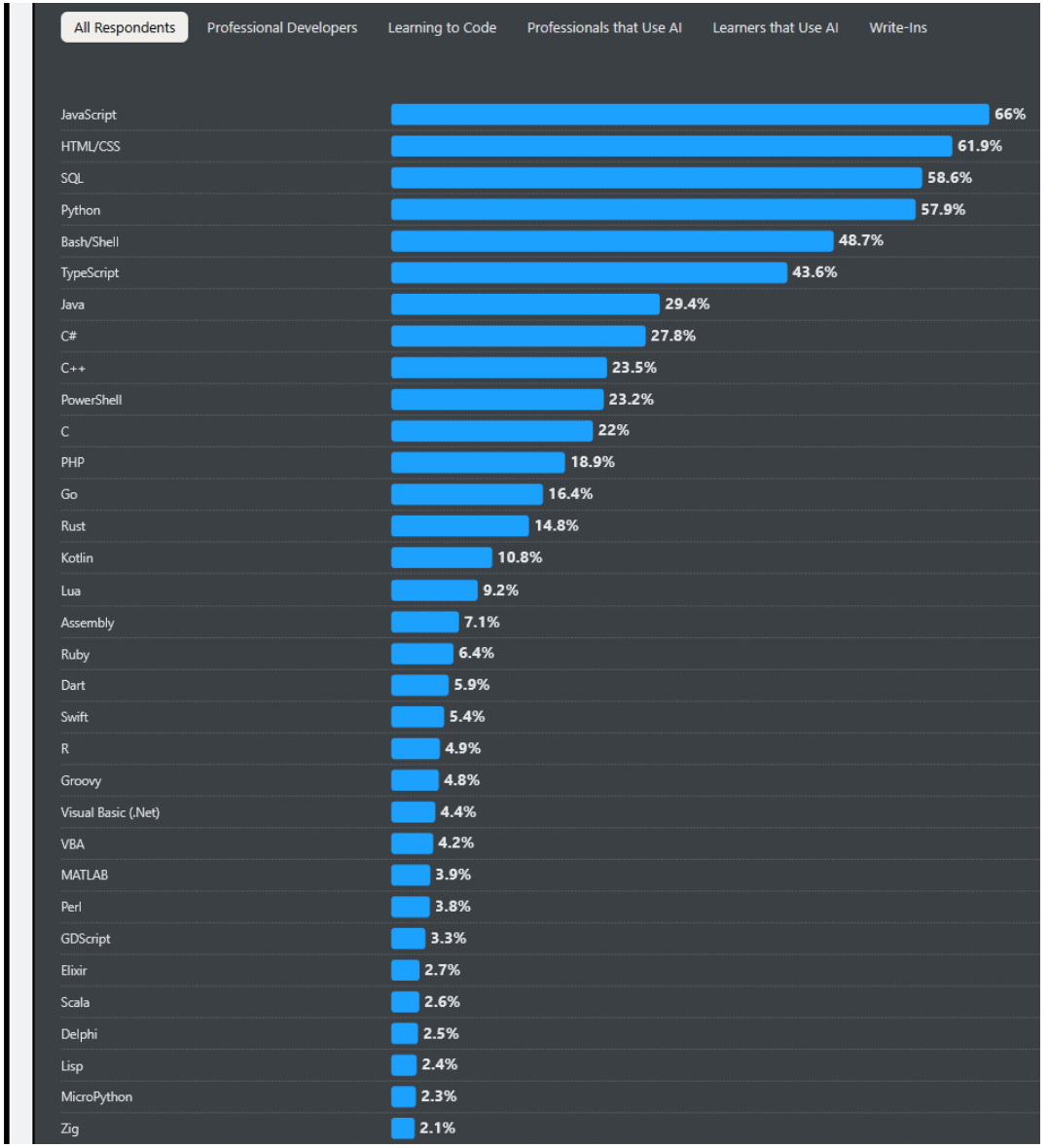
Code documentation and collaboration tools →

„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Sprachen

Programming, scripting, and markup languages

After more than a decade of steady growth, Python's adoption has accelerated significantly. It saw a 7 percentage point increase from 2024 to 2025; this speaks to its ability to be the go-to language for AI, data science, and back-end development.

Which **programming, scripting, and markup languages** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the language and want to continue to do so, please check both boxes in that row.)

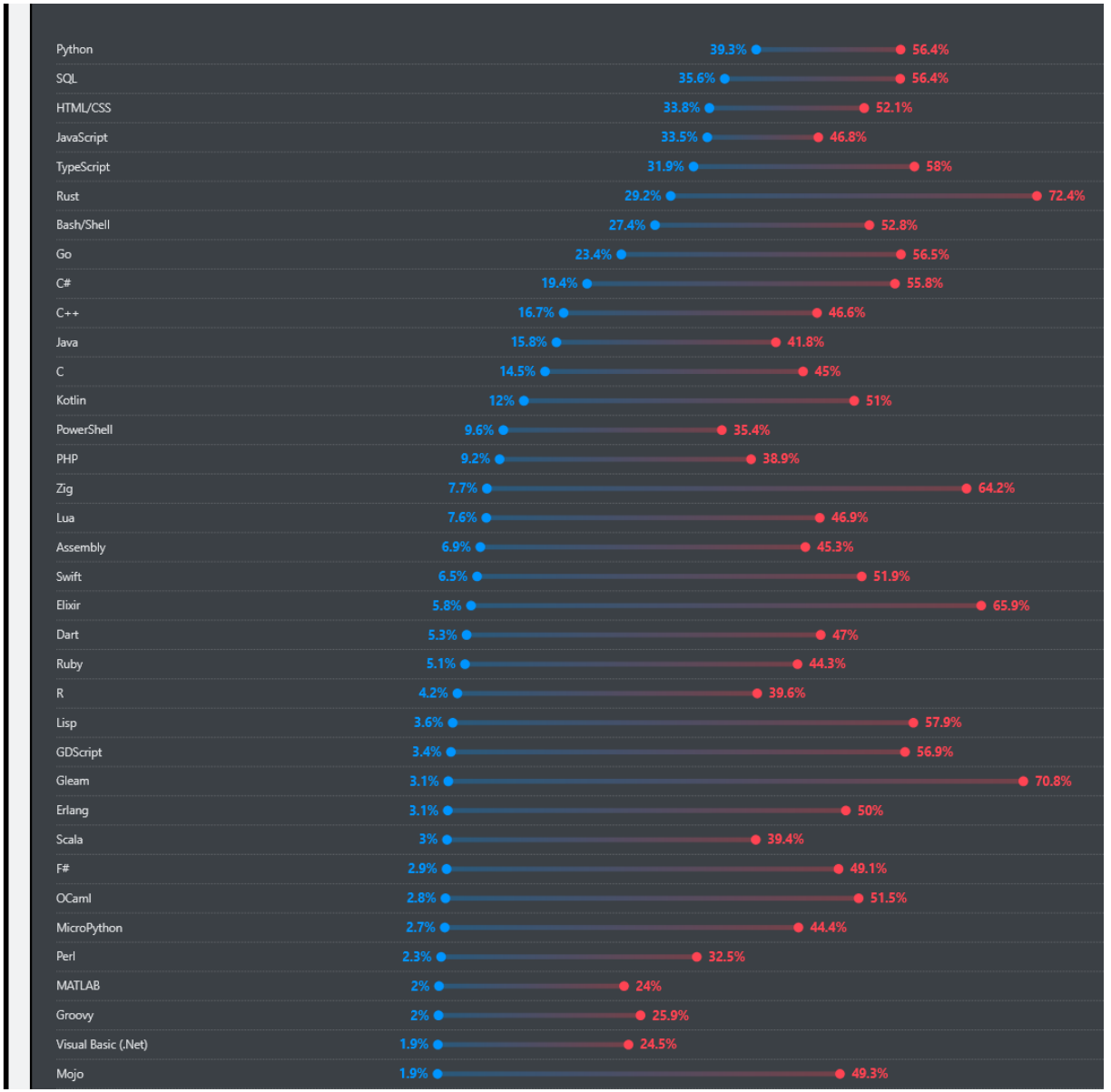


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Sprachen

Programming, scripting, and markup languages

Rust is yet again the most admired programming language (72%), followed by Gleam (70%), Elixir (66%) and Zig (64%). Gleam is a new addition to the list, and for good reason - developers like it!

Which **programming, scripting, and markup languages** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the language and want to continue to do so, please check both boxes in that row.)

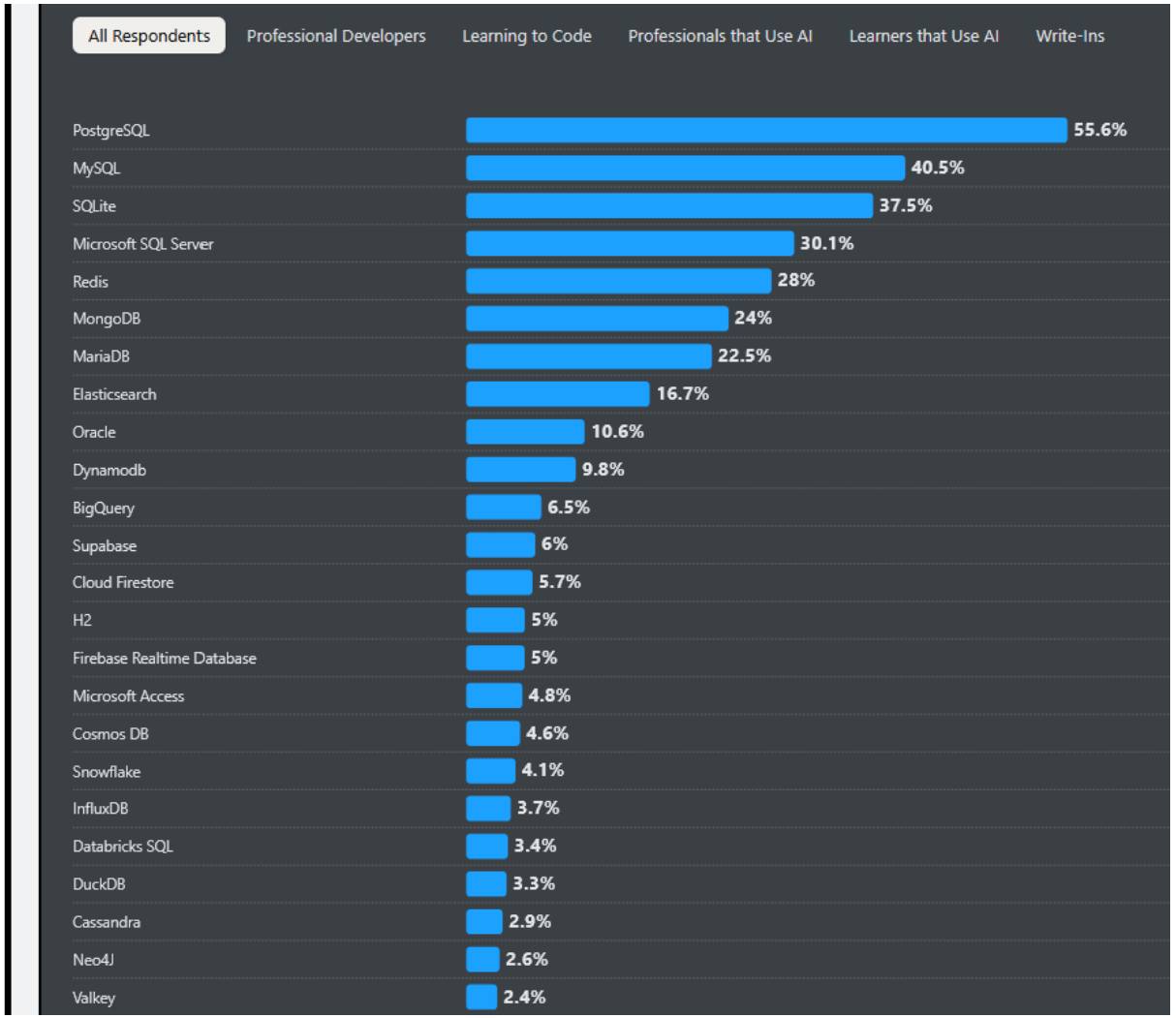


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Datenbanken

Databases

The significant growth in usage for Redis (+8%) highlights its growing importance. As applications become more complex, the need for high-speed, in-memory caching and data structures has made Redis an essential part of the modern tech stack.

Which **database environments** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the database and want to continue to do so, please check both boxes in that row.)

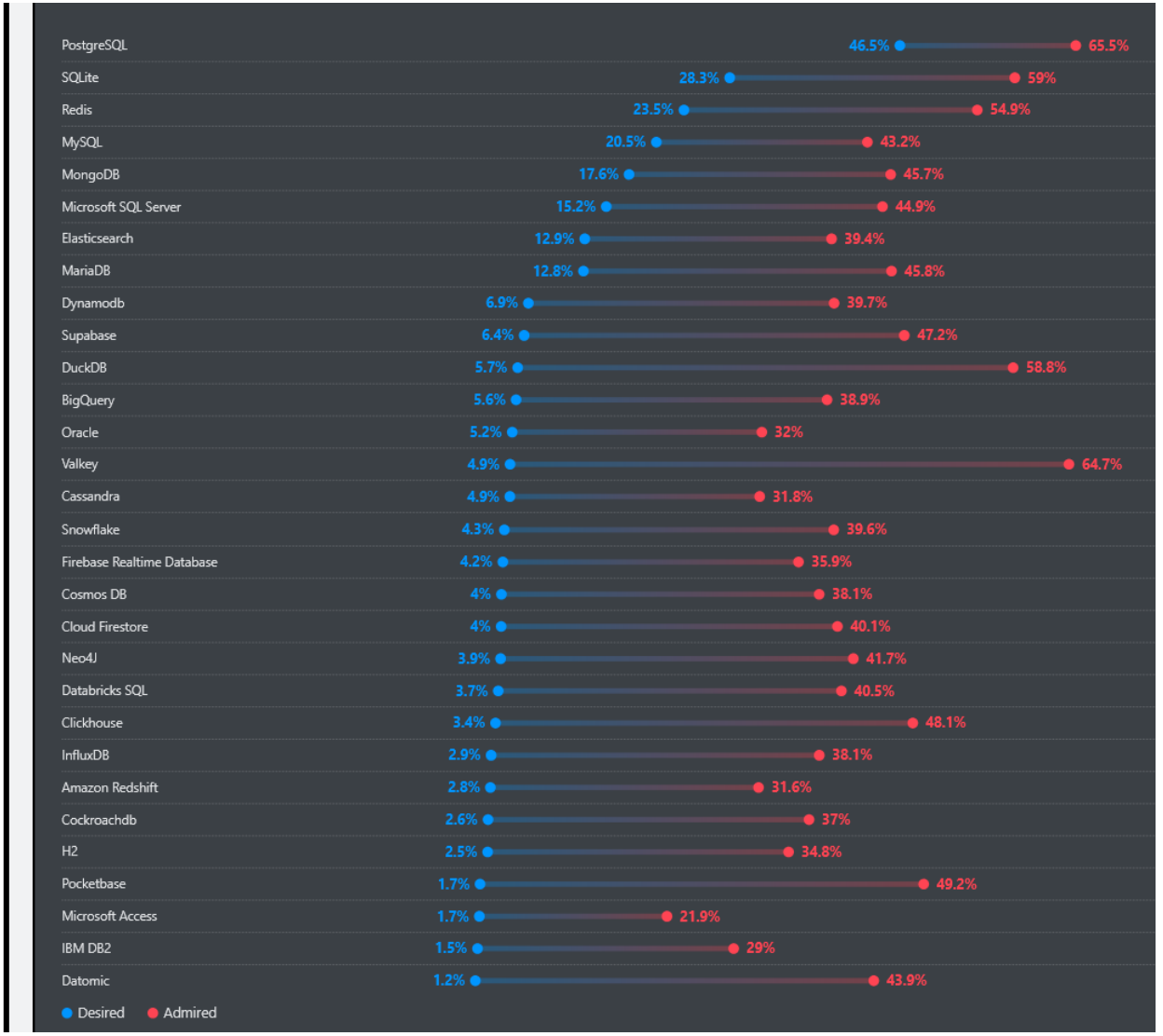


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Datenbanken

Databases

What is it like to be the most desired and the most admired technology in your category? The answer lies with PostgreSQL, ranked highest for both since 2023!

Which **database environments** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the database and want to continue to do so, please check both boxes in that row.)

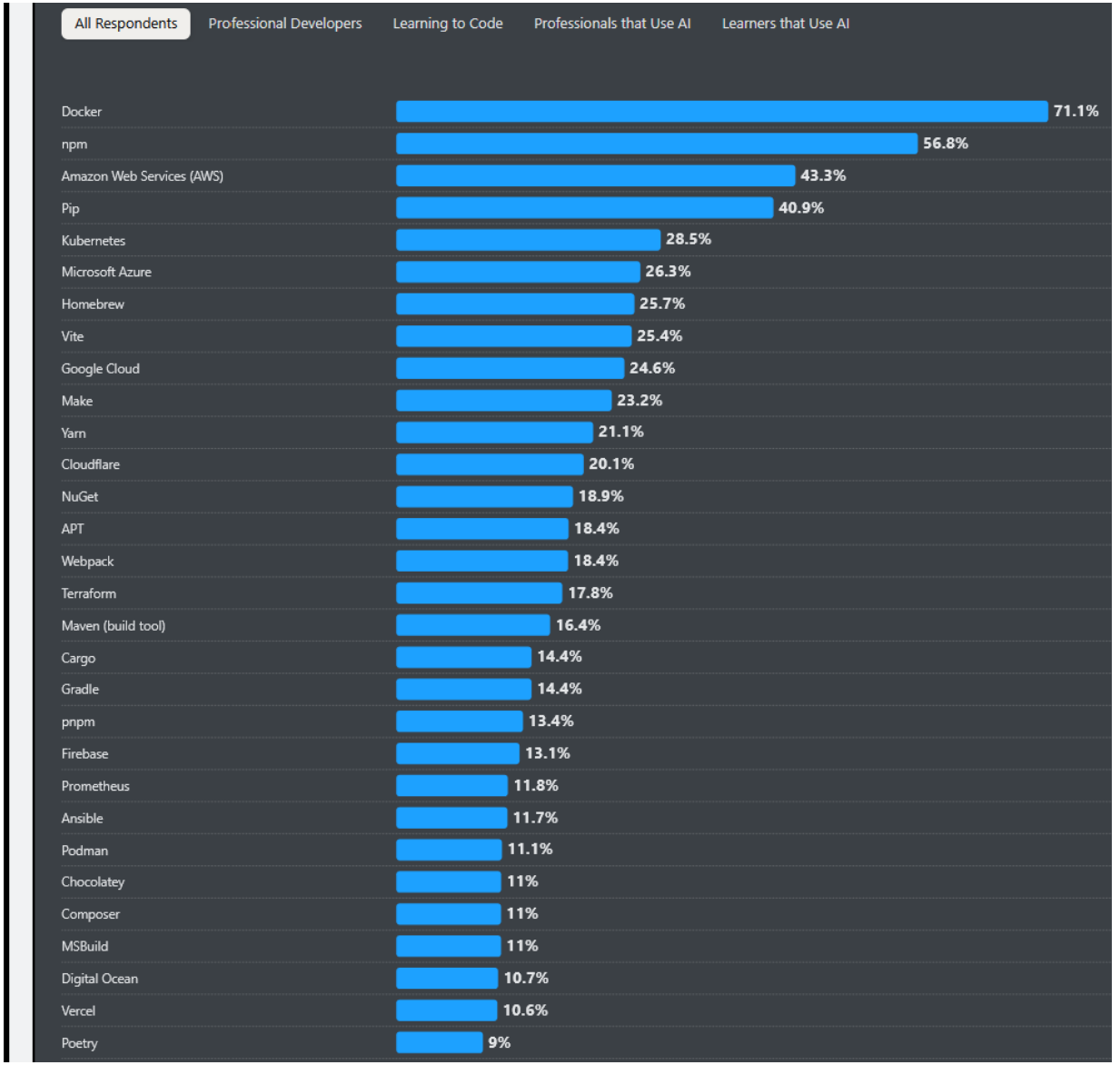


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Cloud Development

Cloud development

Docker has moved from a popular tool to a near-universal one. After years of growth, it experienced a +17 point jump in usage from 2024 to 2025, the largest single-year increase of any technology surveyed. This is partially due to consolidating some technology sectors in this year's survey.

Which **cloud platforms, containerization/orchestration tools, package managers, build tools, and infrastructure as code solutions** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the platform and want to continue to do so, please check both boxes in that row.)

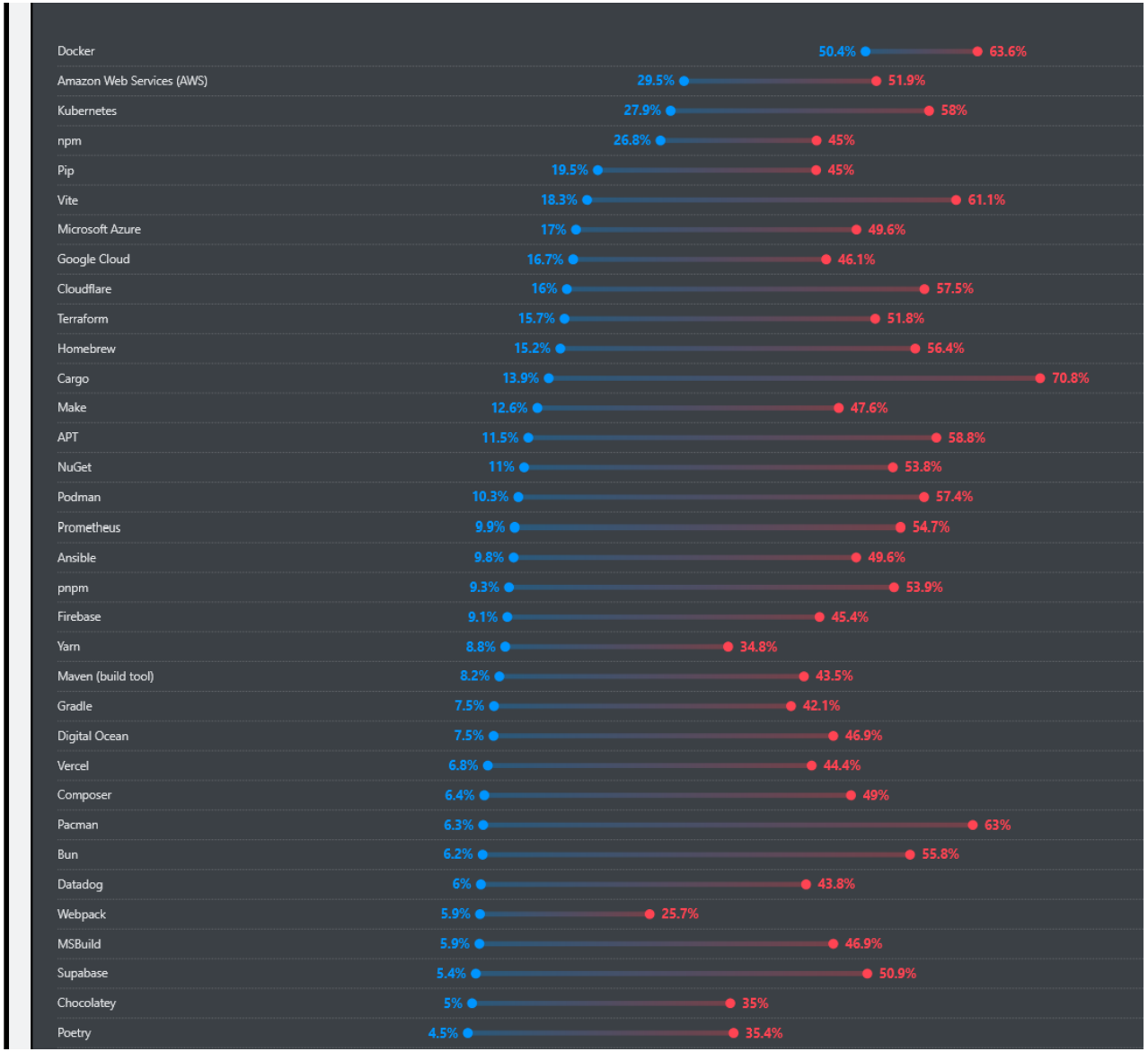


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Cloud Development

Cloud development

Rust's growth is directly tied to the success of its build tool and package manager, Cargo, which is the most admired (71%) cloud development and infrastructure tool this year.

Which **cloud platforms, containerization/orchestration tools, package managers, build tools, and infrastructure as code solutions** have you done extensive development work in over the past year, and which do you want to work in over the next year? (if you both worked with the platform and want to continue to do so, please check both boxes in that row.)

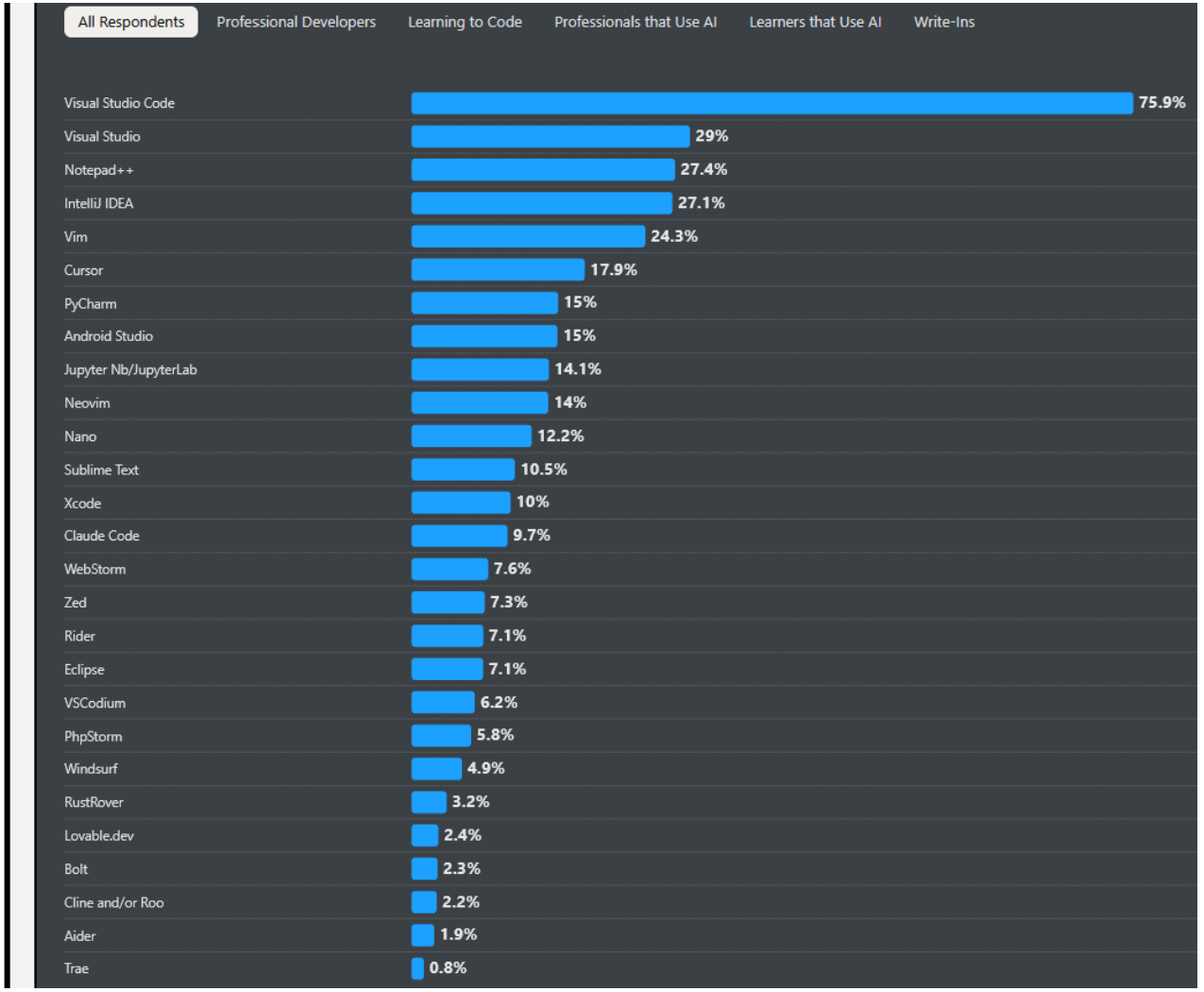


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Dev IDEs

Dev IDEs

Subscription-based, AI-enabled IDEs weren't able to topple the dominance of Visual Studio and Visual Studio Code this year. Both maintained their top spots for the fourth year while relying on extensions as optional, paid AI services.

Which **development environments and AI-enabled code editing tools** did you use regularly over the past year, and which do you want to work with over the next year? Please check all that apply.

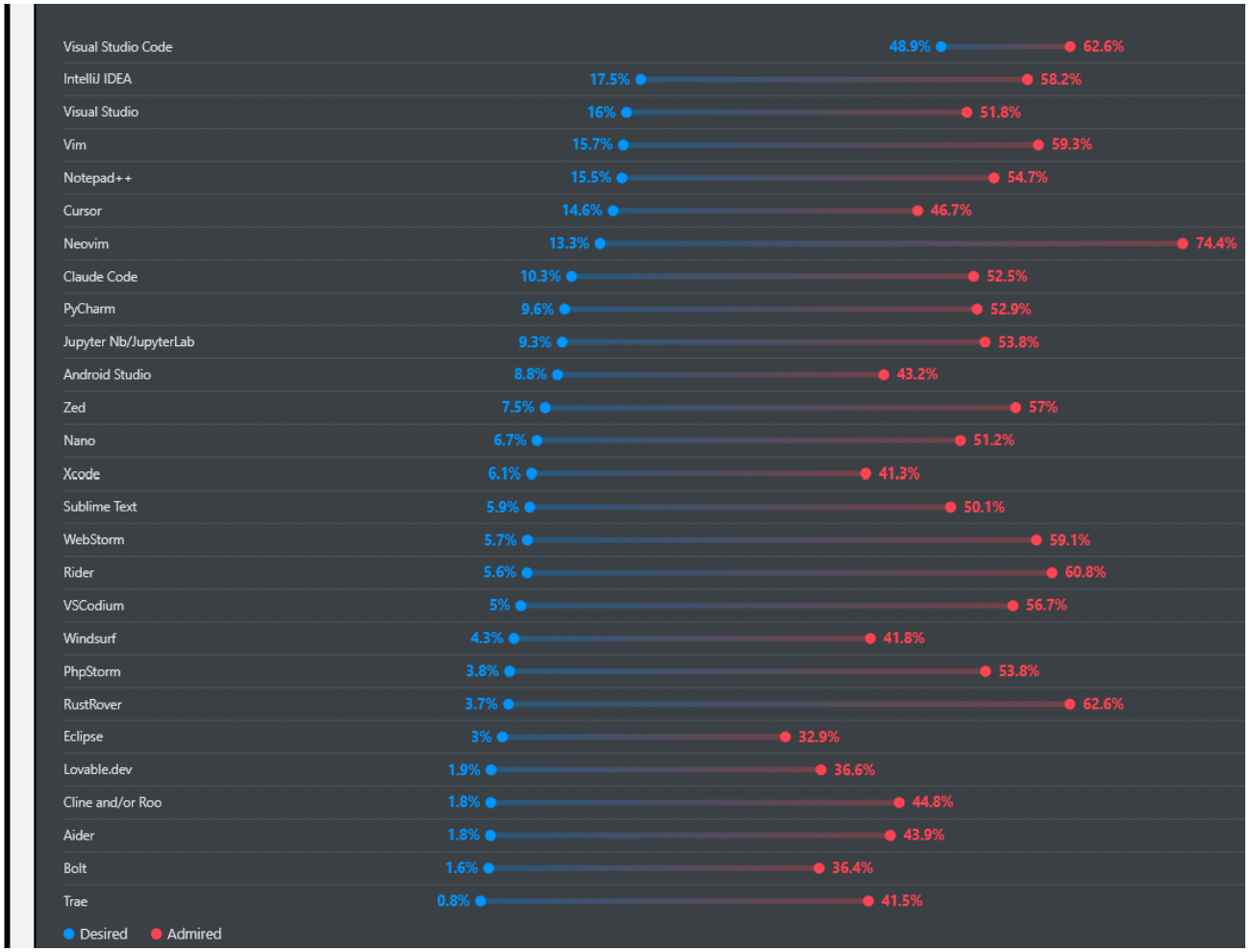


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Dev IDEs

Dev IDEs

Not only is Visual Studio Code the most used IDE for five years in a row, it is also consistently ranked the most desired IDE. Developers that haven't used VS Code overwhelmingly want to use it in the next year.

Which **development environments and AI-enabled code editing tools** did you use regularly over the past year, and which do you want to work with over the next year? Please check all that apply.

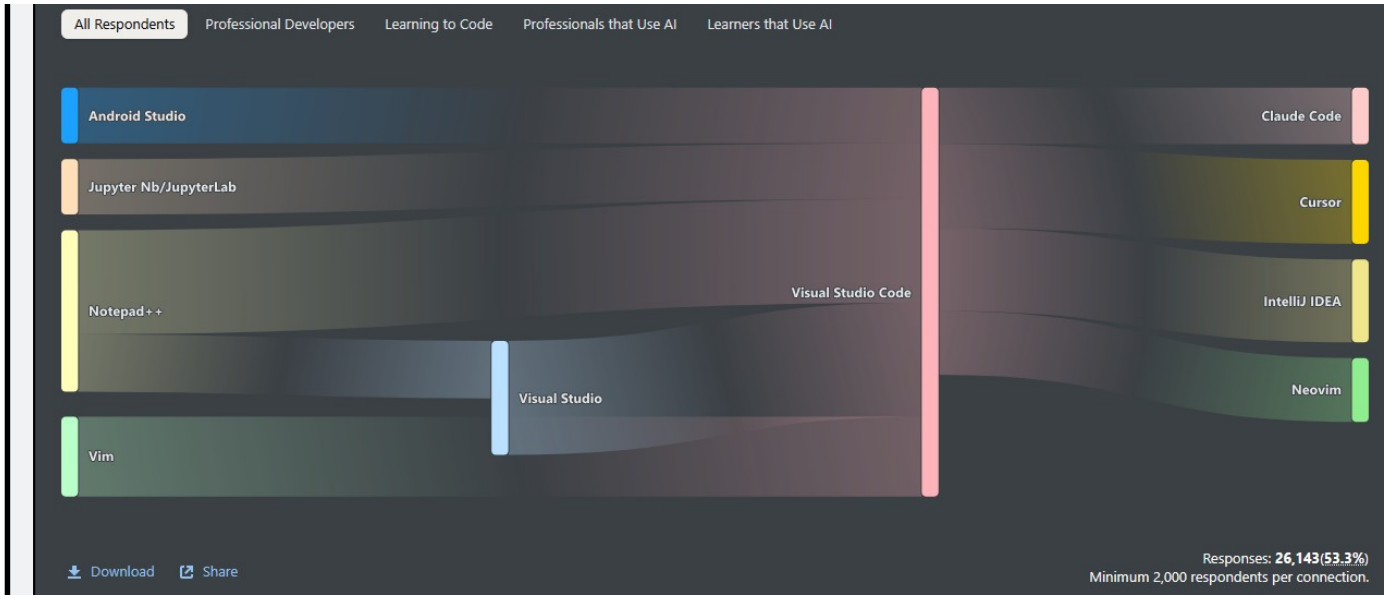


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Dev IDEs

Dev IDEs

The newer AI-enabled editors like Claude Code and Cursor are attracting interest from developers already using the industry standard IDE, VS Code.

Which **development environments and AI-enabled code editing tools** did you use regularly over the past year, and which do you want to work with over the next year? Please check all that apply.

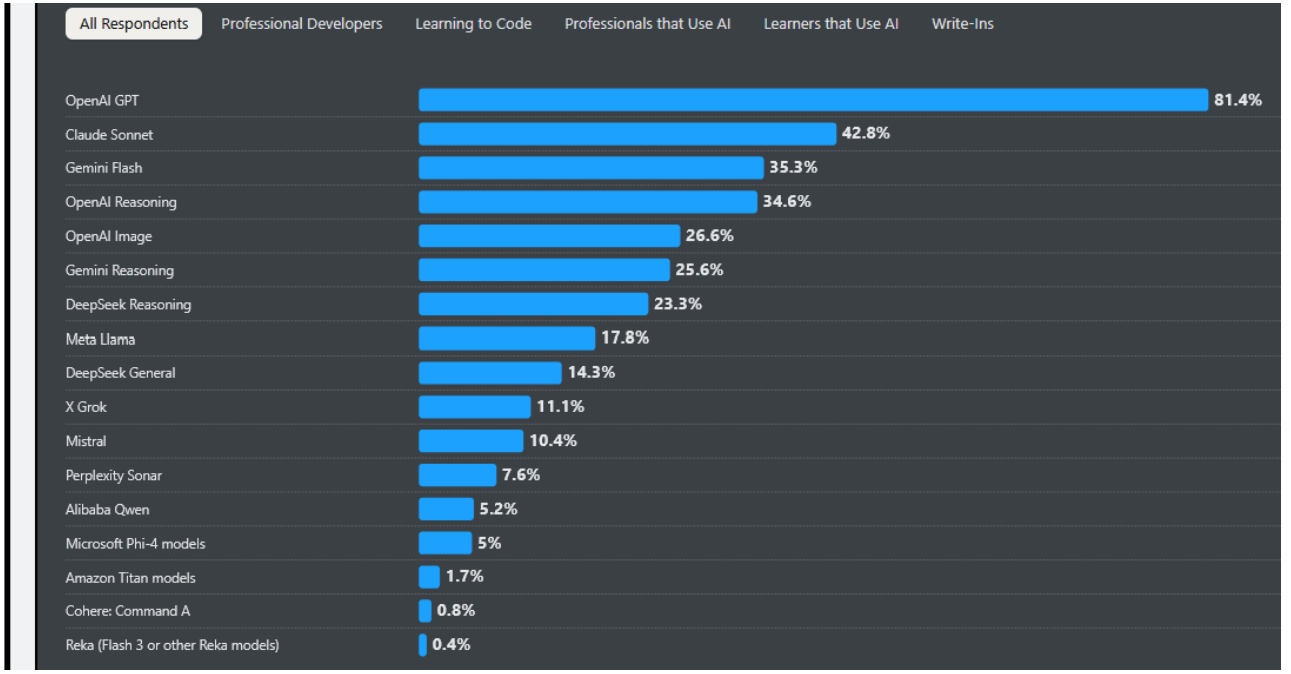


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: LLMs

Large language models

OpenAI's GPT models top the large language model list with 82% of developers indicating they used them for development work in the past year. Anthropic's Claude Sonnet models are used more by professional developers (45%) than by those learning to code (30%).

Which **LLM models for AI tools** have you used for development work in the past year, and which would you like to use next year? Select all that apply.

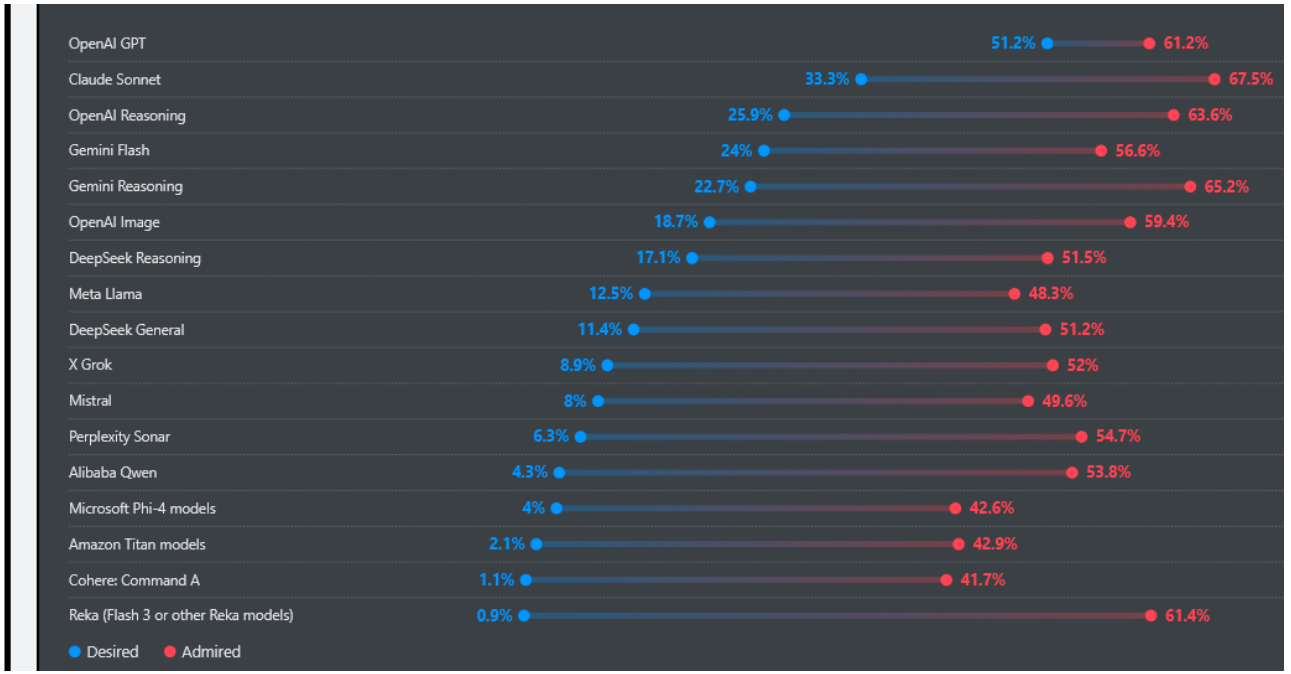


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: LLMs

Large language models

Anthropic's Claude Sonnet is the most admired LLM this year (behind Gemini Reasoning) and second most desired (33%).

Which **LLM models for AI tools** have you used for development work in the past year, and which would you like to use next year? Select all that apply.

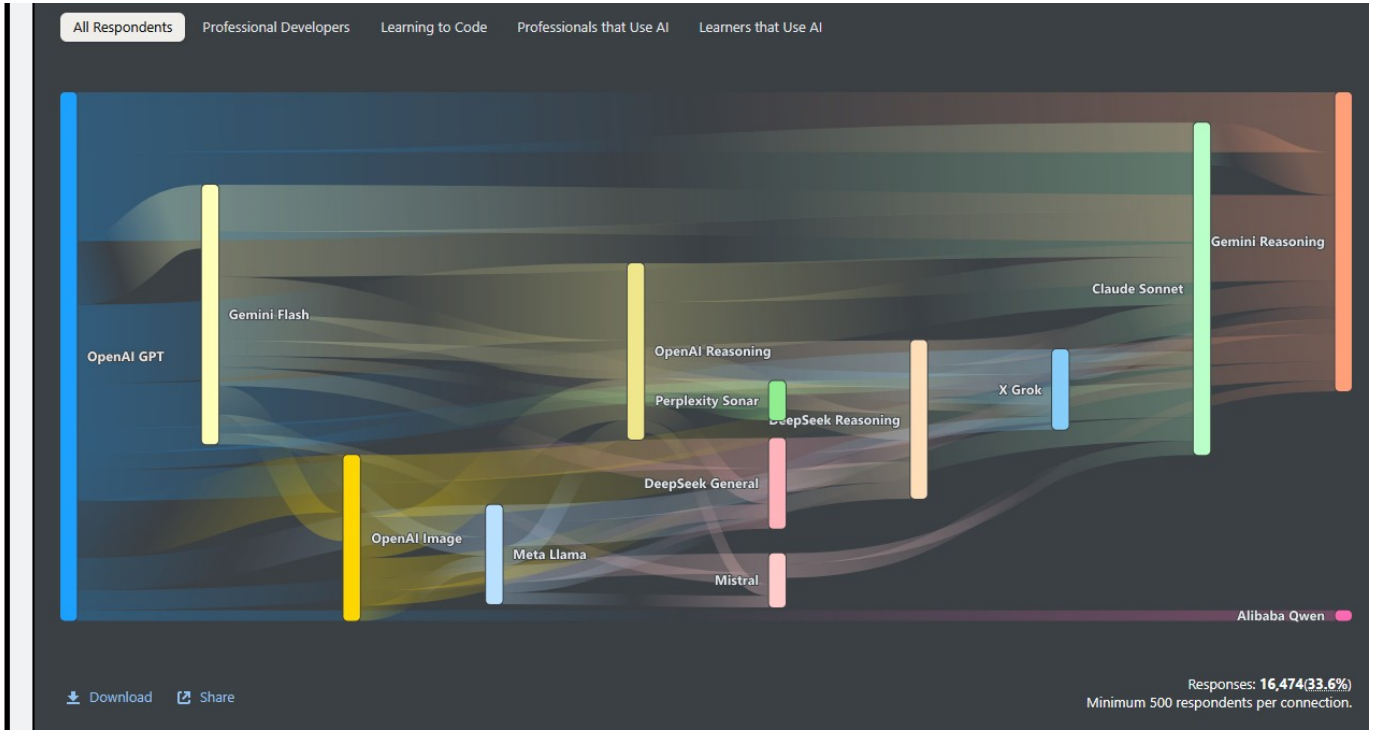


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: LLMs

Large language models

While developers have explored many large language models, OpenAI's GPT models function as the ecosystem's center of gravity. OpenAI GPT users and other LLM users show significant interest in trying Claude Sonnet and Gemini Reasoning models

Which **LLM models for AI tools** have you used for development work in the past year, and which would you like to use next year? Select all that apply.

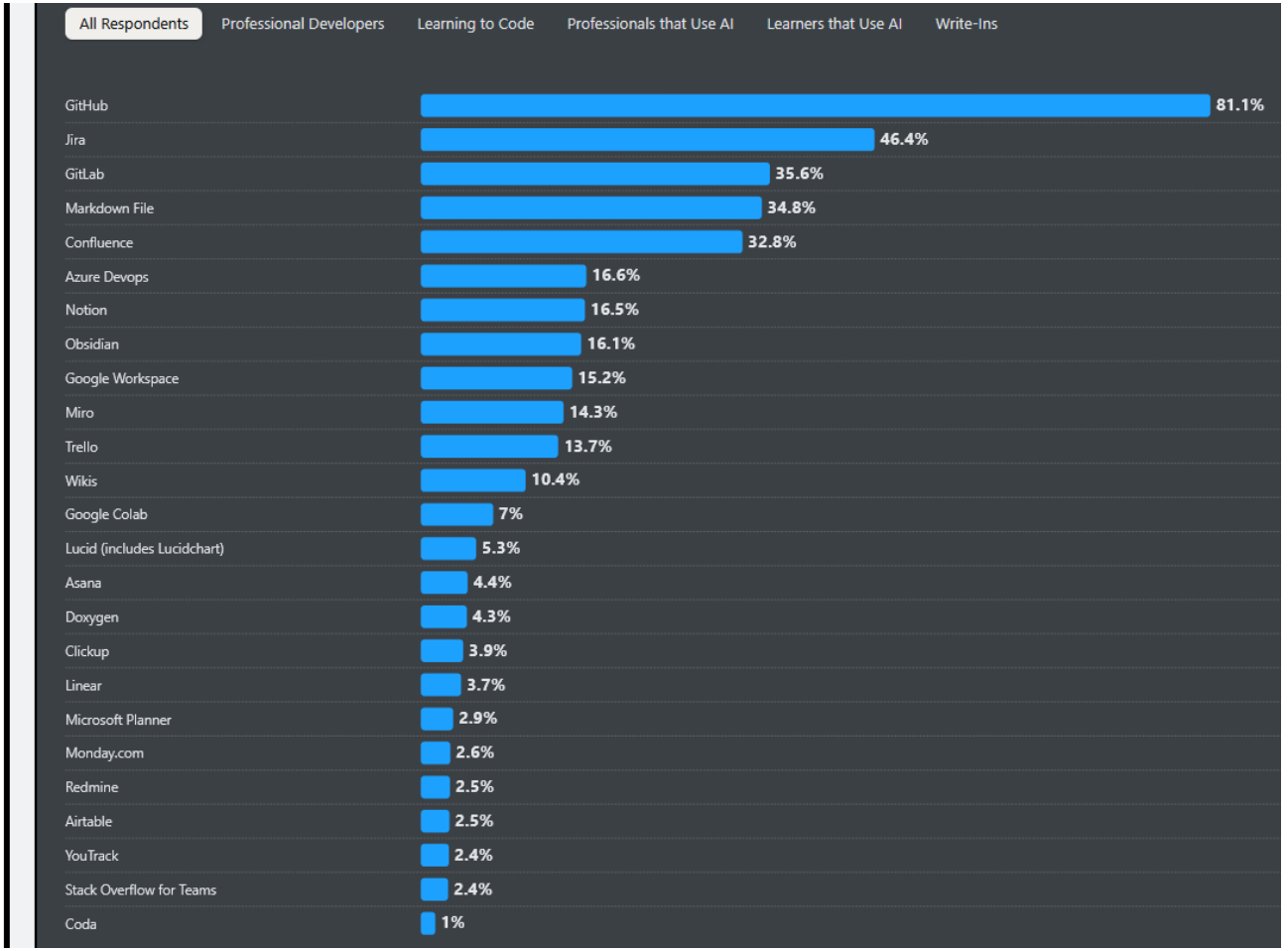


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Code-Doc & Kollab.

Code documentation and collaboration tools

Professional developers that use AI compared to all professional developers show a slight preference for Miro (17% vs. 16%) and Notion (19% vs. 17%) - could this be due to the AI integrations both tools have recently incorporated?

Which **collaborative work management and/or code documentation tools** did you use regularly over the past year, and which do you want to work with over the next year? Select all that apply

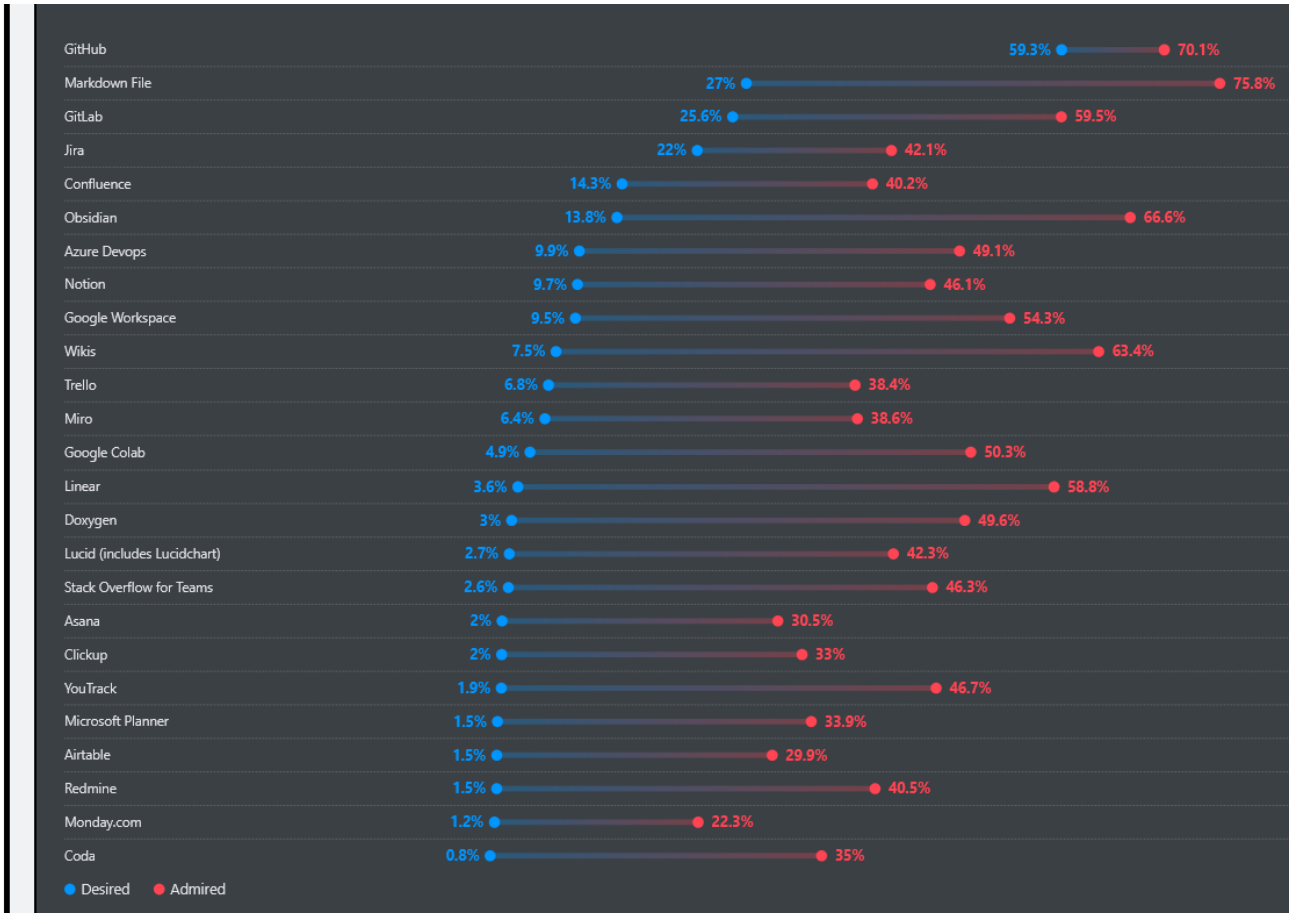


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Code-Doc & Kollab.

Code documentation and collaboration tools

Jira steps down as the most desired tool for code documentation and collaboration and the new top desired tool is GitHub. Markdown continues to be the most admired sync tool for the third year.

Which **collaborative work management and/or code documentation tools** did you use regularly over the past year, and which do you want to work with over the next year? Select all that apply

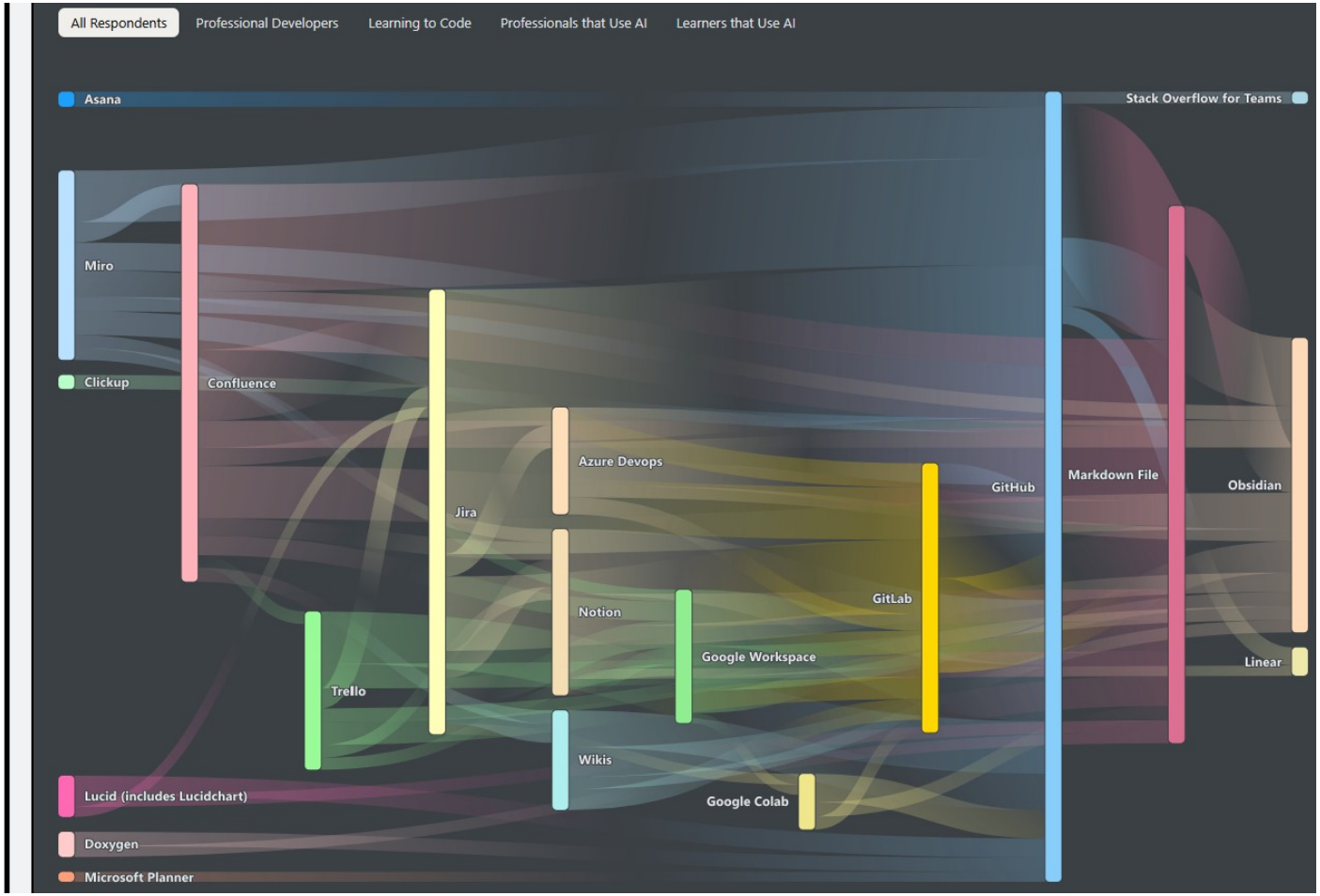


„Aktuelles“ – Stackoverflow Survey 2025 Technologie: Code-Doc & Kollab.

Code documentation and collaboration tools

The data shows the demand to continue using code documentation tools they are currently using is higher for tools that developers adopt for personal knowledge management and documentation, most notably Obsidian, which a significant proportion of Jira and Confluence users that want to use next year.

Which **collaborative work management and/or code documentation tools** did you use regularly over the past year, and which do you want to work with over the next year? Select all that apply

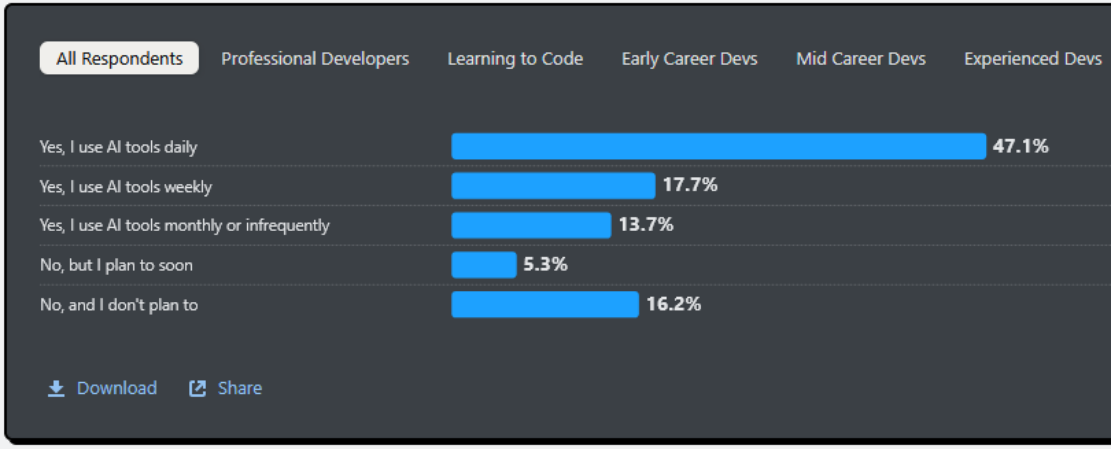


„Aktuelles“ – Stackoverflow Survey 2025
KI: Nutzung und Wahrnehmung

AI tools in the development process

84% of respondents are using or planning to use AI tools in their development process, an increase over last year (76%). This year we can see 51% of professional developers use AI tools daily.

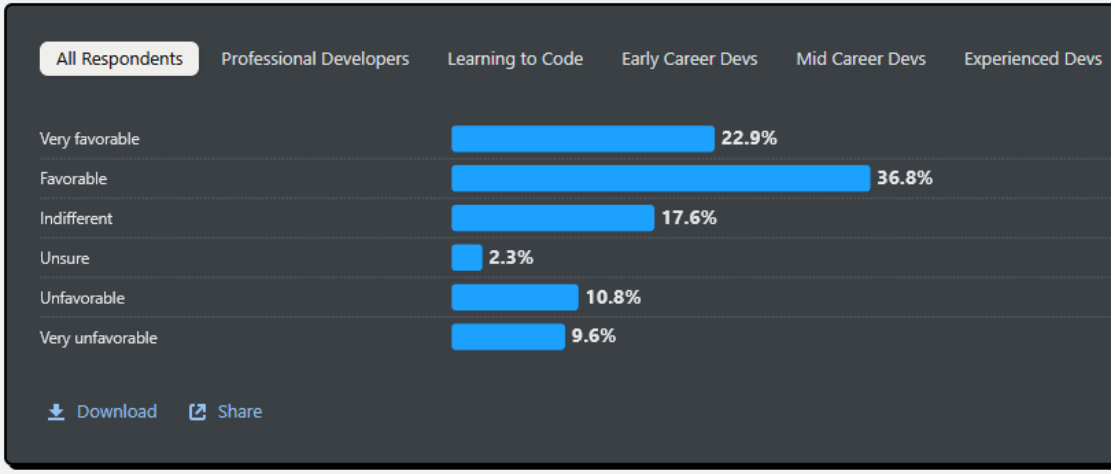
Do you currently use AI tools in your development process?



AI tool sentiment

Conversely to usage, positive sentiment for AI tools has decreased in 2025: 70%+ in 2023 and 2024 to just 60% this year. Professionals show a higher overall favorable sentiment (61%) than those learning to code (53%).

How favorable is your stance on using AI tools as part of your development workflow?

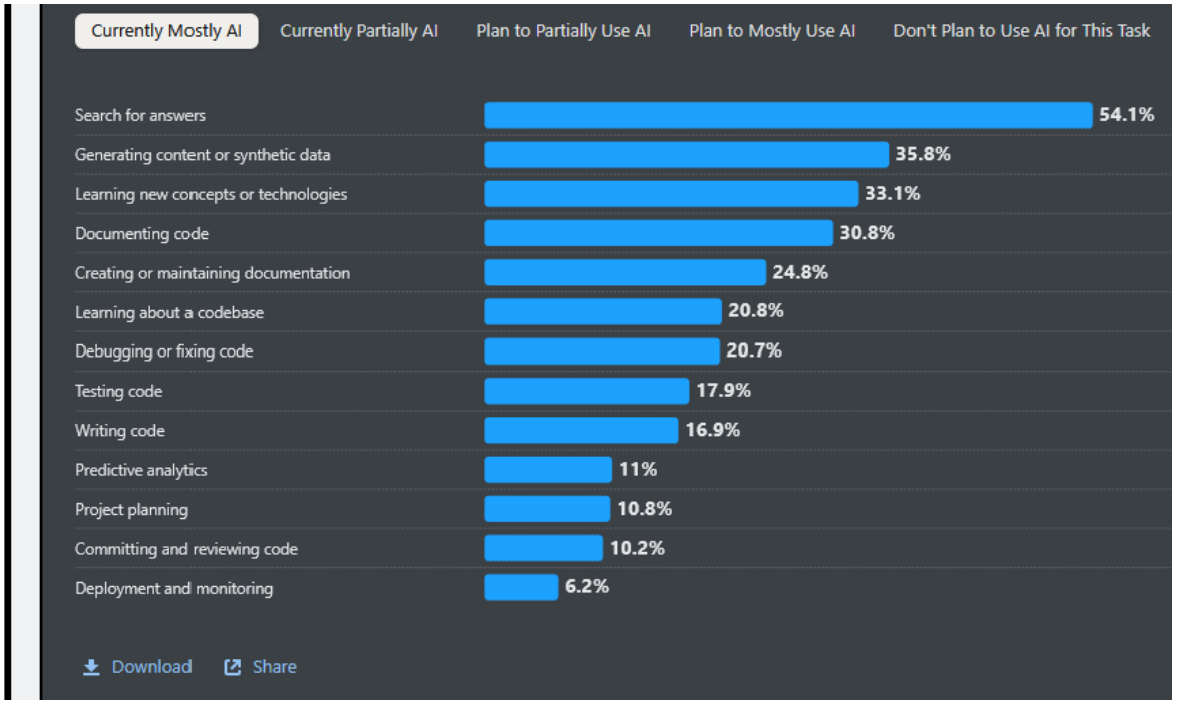


„Aktuelles“ – Stackoverflow Survey 2025 KI: Entwickler-Tools

AI in the development workflow

Developers show the most resistance to using AI for high-responsibility, systemic tasks like Deployment and monitoring (76% don't plan to) and Project planning (69% don't plan to).

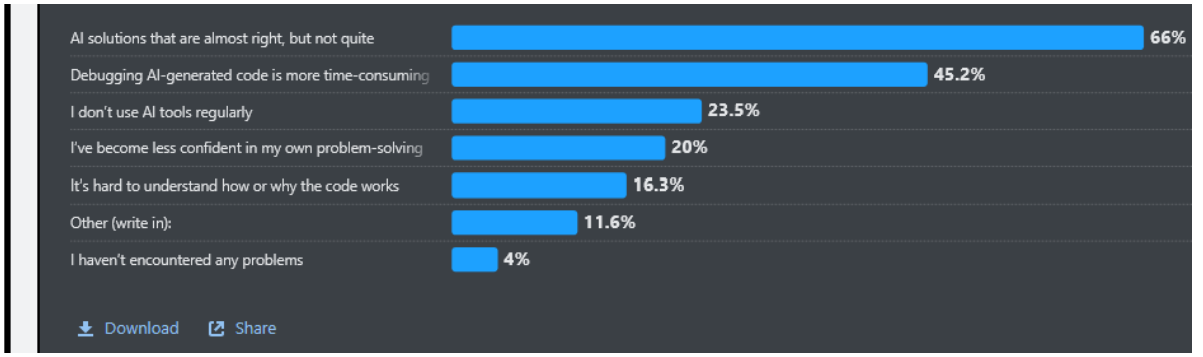
Which parts of your development workflow are you currently integrating into AI or using AI tools to accomplish or plan to use AI to accomplish over the next 3 - 5 years? Please select one for each scenario.



AI tool frustrations

The biggest single frustration, cited by 66% of developers, is dealing with "AI solutions that are almost right, but not quite," which often leads to the second-biggest frustration: "Debugging AI-generated code is more time-consuming" (45%)

When using AI tools, which of the following problems or frustrations have you encountered? Select all that apply.

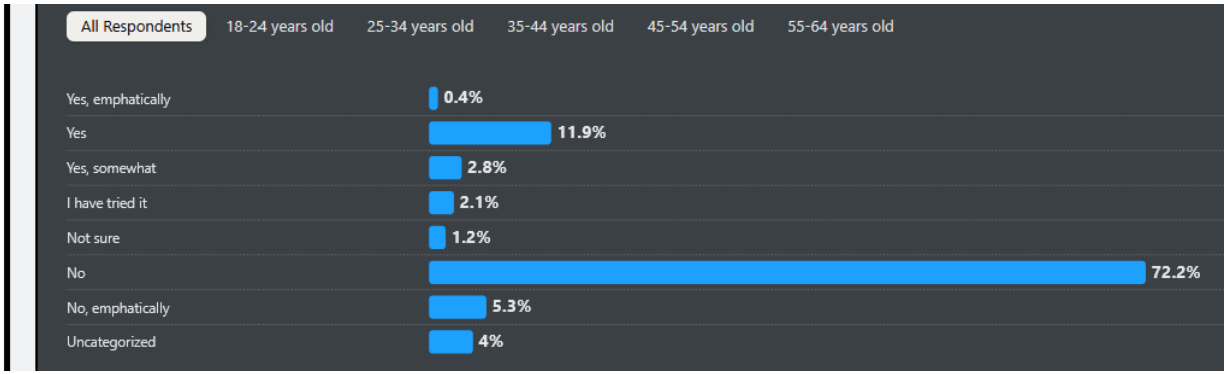


„Aktuelles“ – Stackoverflow Survey 2025 KI: Enwickler-Tools/Agenten

Vibe coding

Most respondents are not vibe coding (72%), and an additional 5% are emphatic it not being part of their development workflow.

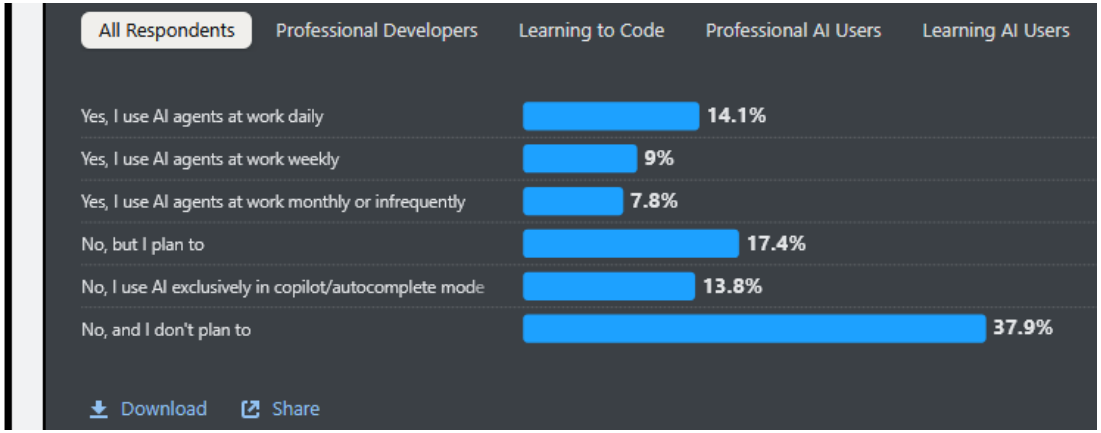
In your own words, is "vibe coding" part of your professional development work? For this question, we define vibe coding according to the Wikipedia definition, the process of generating software from LLM prompts.



AI agents

AI agents are not yet mainstream. A majority of developers (52%) either don't use agents or stick to simpler AI tools, and a significant portion (38%) have no plans to adopt them.

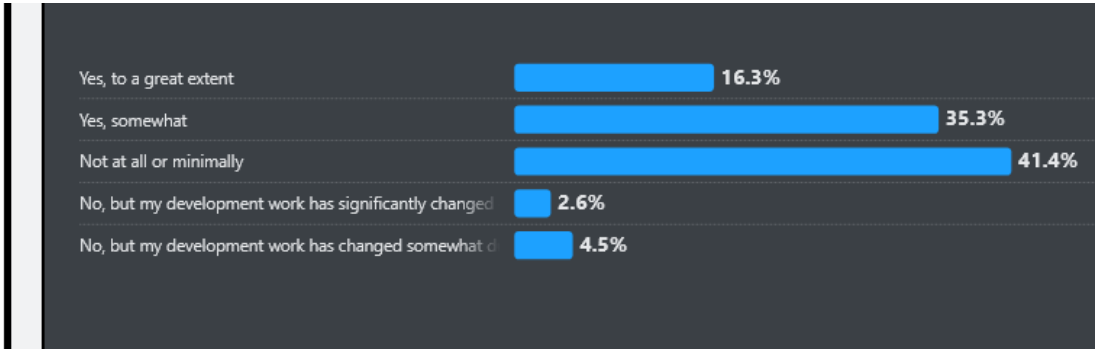
Are you using AI agents in your work (development or otherwise)? AI agents refer to autonomous software entities that can operate with minimal to no direct human intervention using artificial intelligence techniques.



AI agents affect on work productivity

52% of developers agree that AI tools and/or AI agents have had a positive effect on their productivity.

Have AI tools or AI agents changed how you complete development work in the past year?

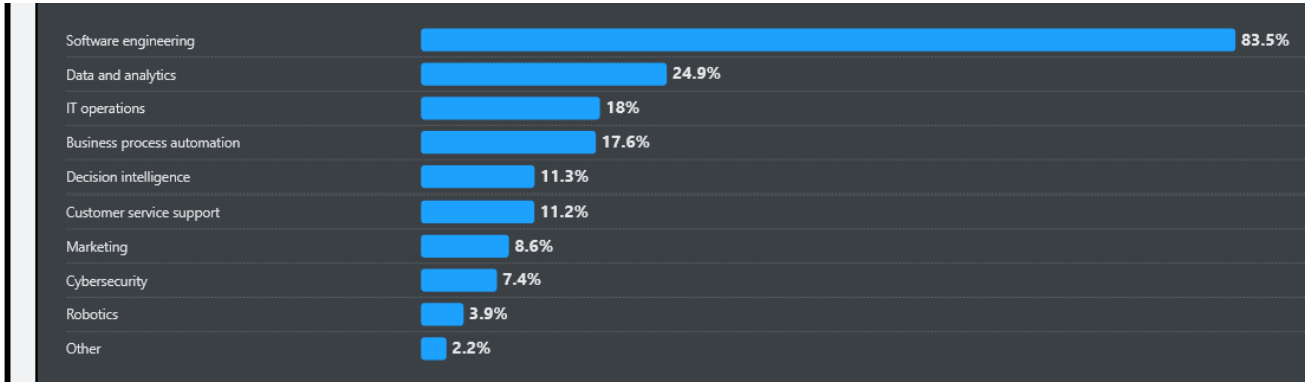


„Aktuelles“ – Stackoverflow Survey 2025 KI: Agenten

AI agent uses at work

If you happen to be using AI agents at work and you are a software developer, chances are high that you are using agents for software development (84%).

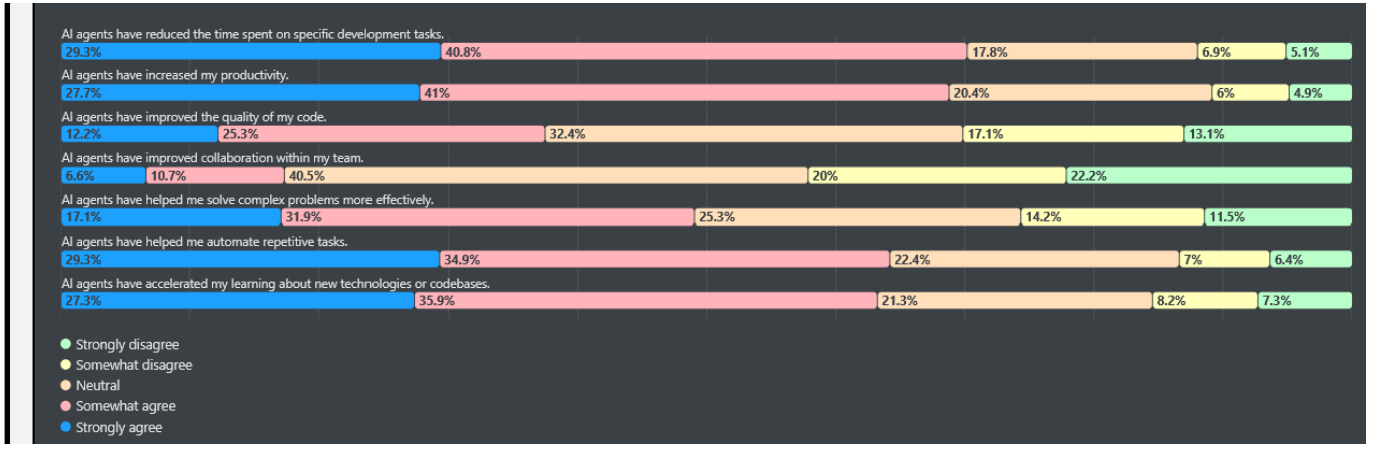
What industry purposes or specific tasks are you using AI agents in your development work? Select all that apply from both lists.



Impacts of AI agents

The most recognized impacts are personal efficiency gains, and not team-wide impact. Approximately 70% of agent users agree that agents have reduced the time spent on specific development tasks, and 69% agree they have increased productivity. Only 17% of users agree that agents have improved collaboration within their team, making it the lowest-rated impact by a wide margin.

To what extent do you agree with the following statements regarding the impact of AI agents on your work as a developer?



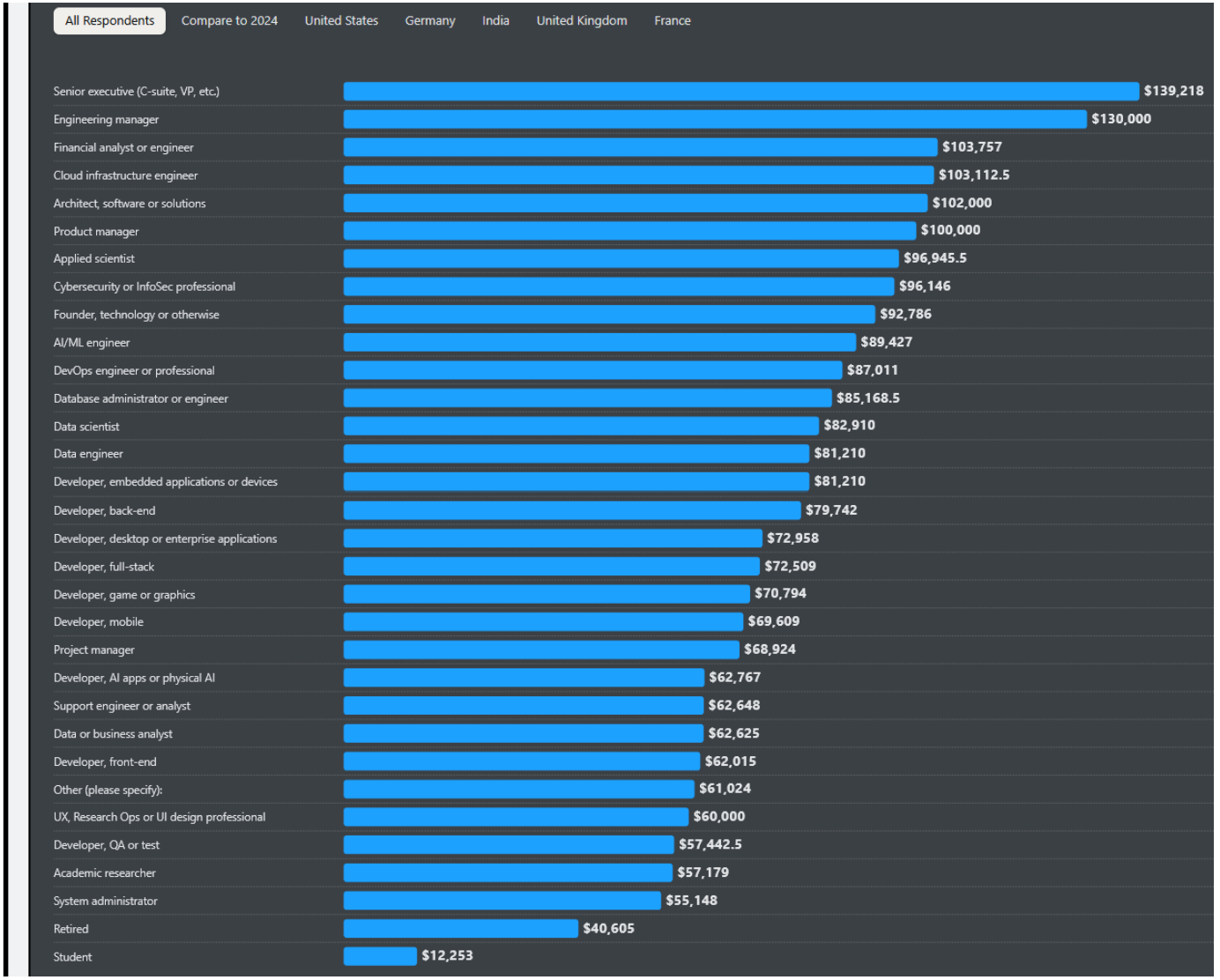
„Aktuelles“ – Stackoverflow Survey 2025 Arbeitsbedingungen: Bezahlung

Salary by developer type

Senior Executives ($139k), Engineering Managers ($130k), and Financial Analysts ($104k) have the highest median annual salaries globally.

The salary gap between the US and other countries is wide for higher paid roles. The median salary for an Engineering Manager in the US is $200,000, compared to $118,000 in Germany and just $52,000 in India.

What is your current total **annual** compensation (salary, bonuses, and perks, before taxes and deductions) in terms of your day-to-day currency? Please enter a whole number in the box below, without any punctuation. If you are paid hourly, please estimate an equivalent yearly salary. If you prefer not to answer, please leave the box empty.



„Aktuelles“ – Neue Richtlinie des NIST für Passwort-Regeln

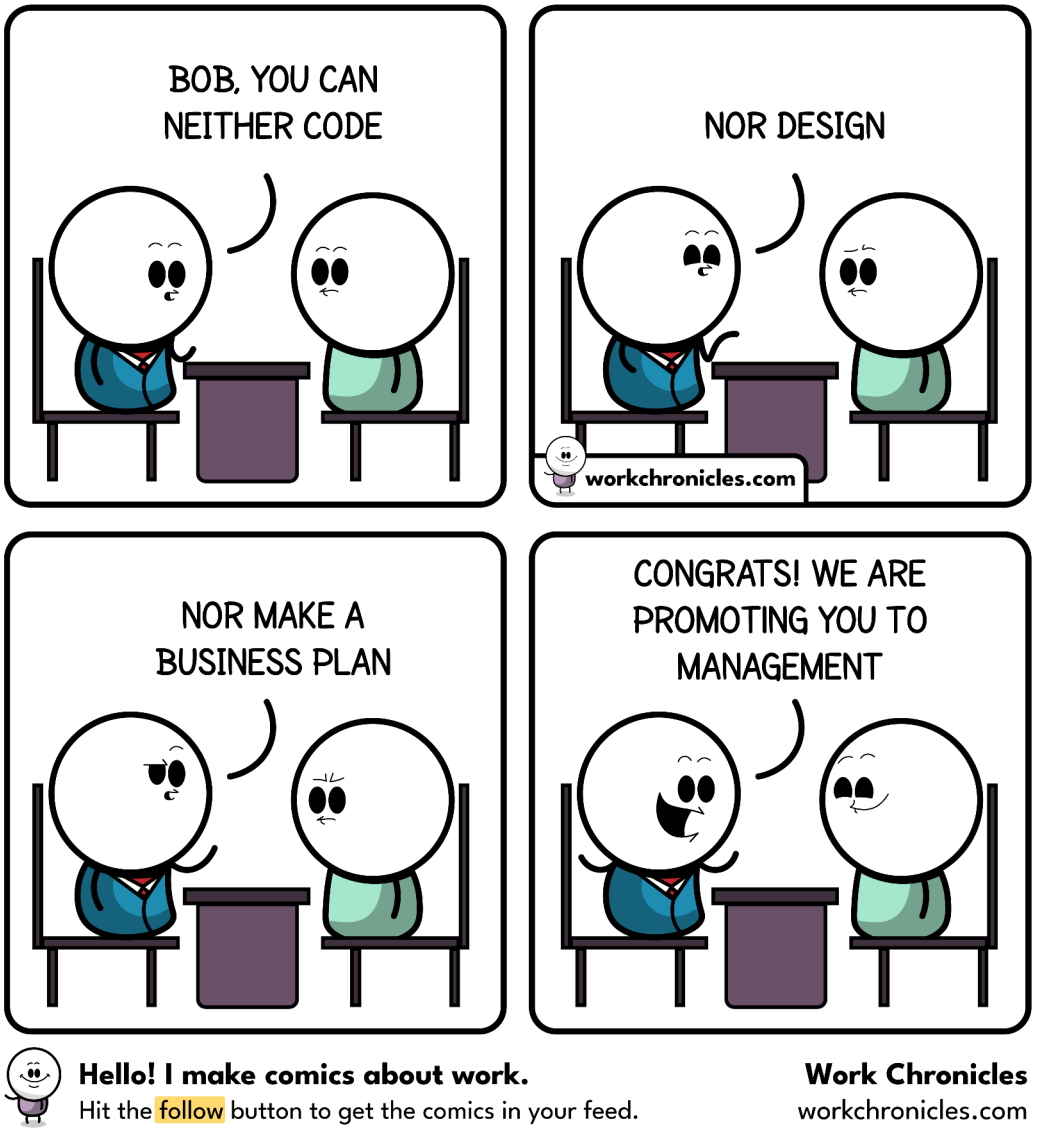
- Das NIST (National Institute of Standards and Technology) veröffentlicht (regelmäßig) Regeln und Empfehlungen zur Passwort-Hygiene, die Firmen implementieren und sich dann konform erklären können
- In den aktuellen Empfehlungen sind endlich die umstrittensten Regeln auch „verboten“

„Aktuelles“ – Neue Richtlinie des NIST für Passwort-Regeln

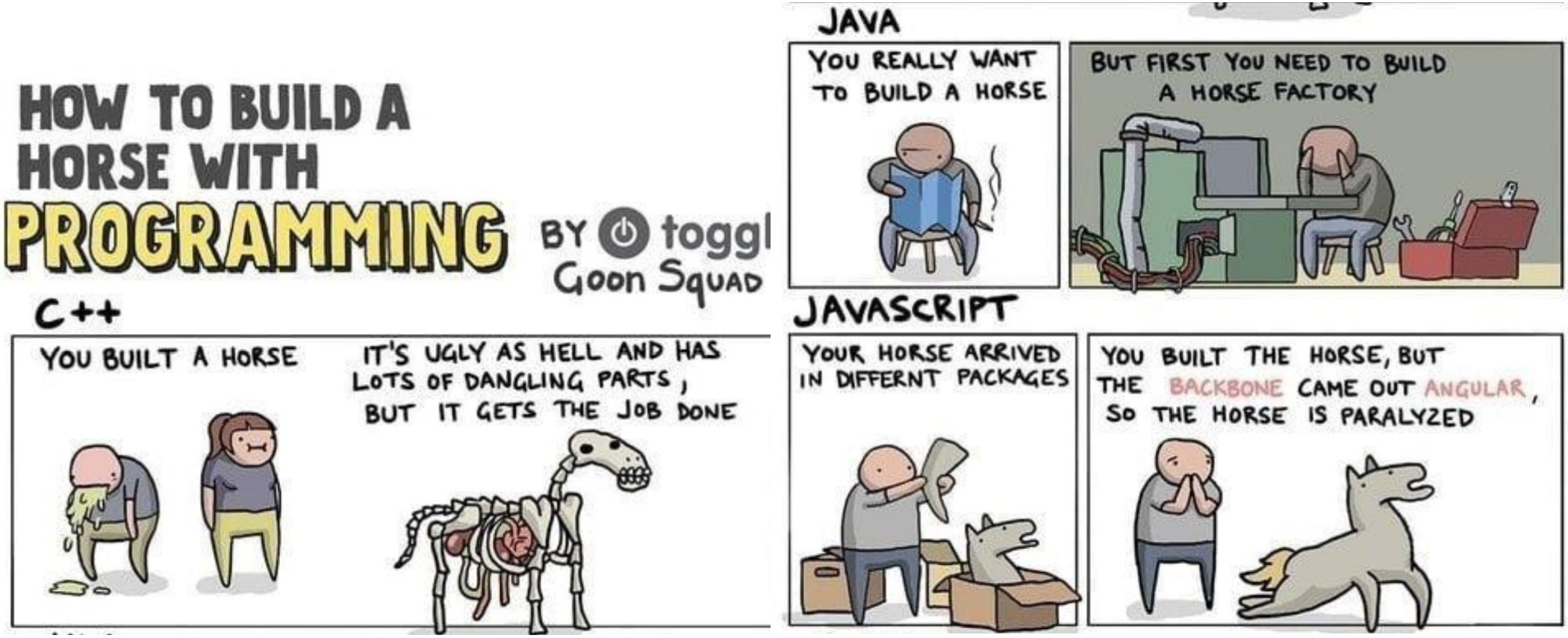
How have the NIST password requirements changed?

	Old NIST password guidelines	New NIST password guidelines
Password length	Limit to 8-16 characters	Longer passwords up to 64 characters
Character complexity	Encouraged	Not required
Mandatory password changes	Required monthly	Only when compromised
Password blocklist	Basic terms	Breached passwords, patterns, and common variations
Recovery methods	Security questions	Links and verification codes
Additional precautions	–	MFA and password managers

„Aktuelles“ – Soft Skills sind in der IT wichtig … insbesondere in dieser Rolle ;)



„Aktuelles“ – … zum Schmunzeln



„Aktuelles“ – … zum Schmunzeln

Fahrplan für die Vorlesung

1. Agile Methoden FG
2. Anforderungsmanagement FG
3. Software-Architektur FG
4. Software-Engineering JD
5. Einführung in den Softwareentwicklungsprozeß JD
6. Versionsverwaltung JD
7. Continuous Integration/Contin. Delivery (CI/CD) JD
8. Software Craftsmanship JD
9. Domain-Driven-Design (DDD)/SW-Architektur ??? JD

FG – Florian Glufke, JD – Janko Dietzsch

1. Software-Engineering

- Engl. Wikipedia:
  - „Software engineering is the study and an application of engineering to the design, development, implementation and maintenance of software in a systematic method.“
- Deut. Wikipedia:
  - „Die Softwaretechnik ist eine deutschsprachige Übersetzung des engl. Begriffs Software Engineering und beschäftigt sich mit der Herstellung oder Entwicklung von Software, der Organisation und Modellierung der zugehörigen Datenstrukturen und dem Betrieb von Softwaresystemen.“
  - „Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.“ (Balzert, *Lehrbuch der Software-Technik Bd. 1)*

Kernprozesse

1. Planung

- Anforderungserhebung
- Lastenheft (Anforderungsdefinition)
- Pflichtenheft (Mit technischen Ansätzen verfeinertes Lastenheft)
- Aufwandsschätzung (z. B. mittels Function-Point-Verfahren oder COCOMO)
- Vorgehensmodell

2. Analyse

- Auswertung
- Mock-up
- Prozessanalyse / Prozessmodell
- Systemanalyse
- Strukturierte Analyse (SA)
- Objektorientierte Analyse (OOA)

3. Entwurf

- Softwarearchitektur
- Strukturiertes Design (SD)
- Objektorientiertes Design (OOD)
- Fundamental Modeling Concepts (FMC)

4. Programmierung

- Normierte Programmierung
- Strukturierte Programmierung
- Objektorientierte Programmierung (OOP)
- Funktionale Programmierung

5. Validierung und Verifikation

- Modultests (Low-Level-Test)
- Integrationstests (Low-Level-Test)
- Systemtests (High-Level-Test)
- Akzeptanztests (High-Level-Test)

Unterstützungsprozesse

6. Anforderungsmanagement

7. Projektmanagement

- Risikomanagement
- Projektplanung
- Projektverfolgung und -steuerung
- Management von Lieferantenvereinbarungen

8. Qualitätsmanagement

- Capability Maturity Model
- Spice (Norm) (Software Process Improvement and Capability Determination)
- Incident Management
- Problem Management
- Softwaremetrik (Messung von Softwareeigenschaften)
- statische Analyse (Berechnung von Schwachstellen)
- Softwareergonomie

9. Konfigurationsmanagement

- Versionsverwaltung
- Änderungsmanagement / Veränderungsmanagement
- Releasemanagement
- Application-Management (ITIL)

10. Softwareeinführung

11. Dokumentation

- Technische Dokumentation
- Softwaredokumentation
- Systemdokumentation (Weiterentwicklung und Fehlerbehebung)
- Betriebsdokumentation (Betreiber/Service)
- Bedienungsanleitung (Anwender)
- Geschäftsprozesse (Konzeption der Weiterentwicklung)
- Verfahrensdokumentation (Beschreibung rechtlich relevanter Softwareprozesse)

Software-Engineering – Art vs. Science ...

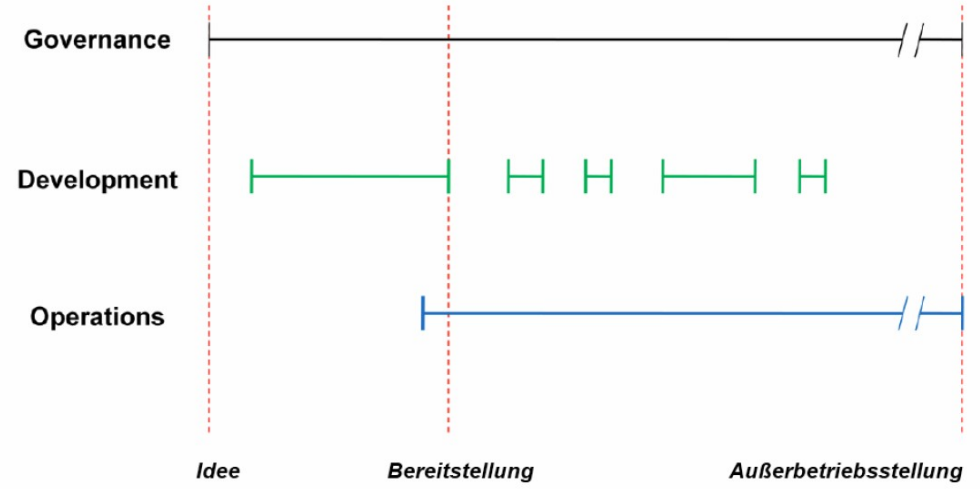
- Science:
  - Algorithmen, Graphentheorie, Datenstrukturen, …
  - Numerische Algorithmen, FFT, …
  - ...
- Engineering:
  - Prozesse, Werkzeuge, Metriken, ...
  - ...
- Art:
  - Design - User-Interface (UI), User-Experience (UX)
  - Marketing
  - ...
- Mix aus allen drei definiert schließlich den Erfolg
  - Möglichst alle Aspekte berücksichtigen
  - Möglichst ein „vielfältiges Team“, möglichst viele verschiedene Talente

Online Quellen

- Google'n, Bing'en, DuckDuckGo'en oder ChatGPT'en ...
  - Blogs, Artikel, … etc
  - Podcasts:
    - Software Enginnering Radio: http://www.se-radio.net
    - Software ArchitekTOUR-Podcast von Heise Developer: https://www.heise.de/developer/SoftwareArchitekTOUR-4076349.html
    - Etwas mehr .NET-lastig: https://www.dotnetrocks.com/
    - Hanselminutes: http://hanselminutes.com/
    - Software Engineering Daily: http://softwareengineeringdaily.com/category/podca
    - …
  - Videocasts:
    - Zum Teil gibt es die obigen Podcasts auch als Videocast auf Youtube
    - Diverses weites Feld in der „Youtube-University“ ...
- MOOC's:
  - edX: https://www.edx.org/
  - Coursera: https://www.coursera.org/
  - Udacity: https://www.udacity.com/
  - ...

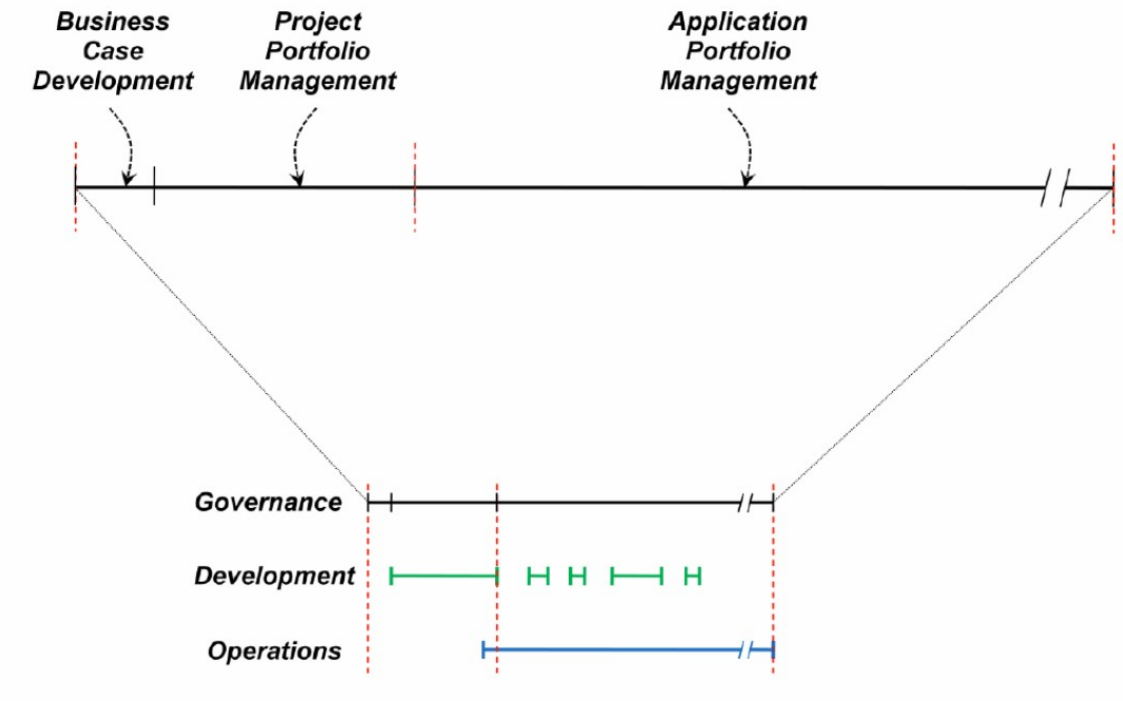
2. Application Lifecycle-Management (ALM)

- Hier nur als grober Leitfaden zur Einordnung der später folgenden Themen (orientiert an Chappell, *What is ALM, 2010*)
- Mehr als der Software Development Lifecycle (SDLC) – ***kompletter Lebenszyklus*** einer Anwendung
- Drei gut identifizierbare Ebenen:
  - Governance (Steuerung)
  - Development (Entwicklung)
  - Operation (Betrieb)
- Geprägt von Ereignissen:
  - Idee
  - Breitstellung
  - Außerbetriebsstellung



2.1 Governance

- Erstreckt sich über den kompletten Lebenszyklus der Anwendung
- Stellt sicher, das die Erwartungen der Anwender und der Organisation erfüllt werden:
  - Wirtschaftlichkeitsbetrachtung
  - Projektmanagement für den Entwicklungsprozess
  - Übergang in das aktive Portfolio des Unternehmens
  - Upgrades & Updates aus geschäftlicher Sicht betrachten & planen
  - „Koordiniertes Ableben“ am Ende des Produktzyklus

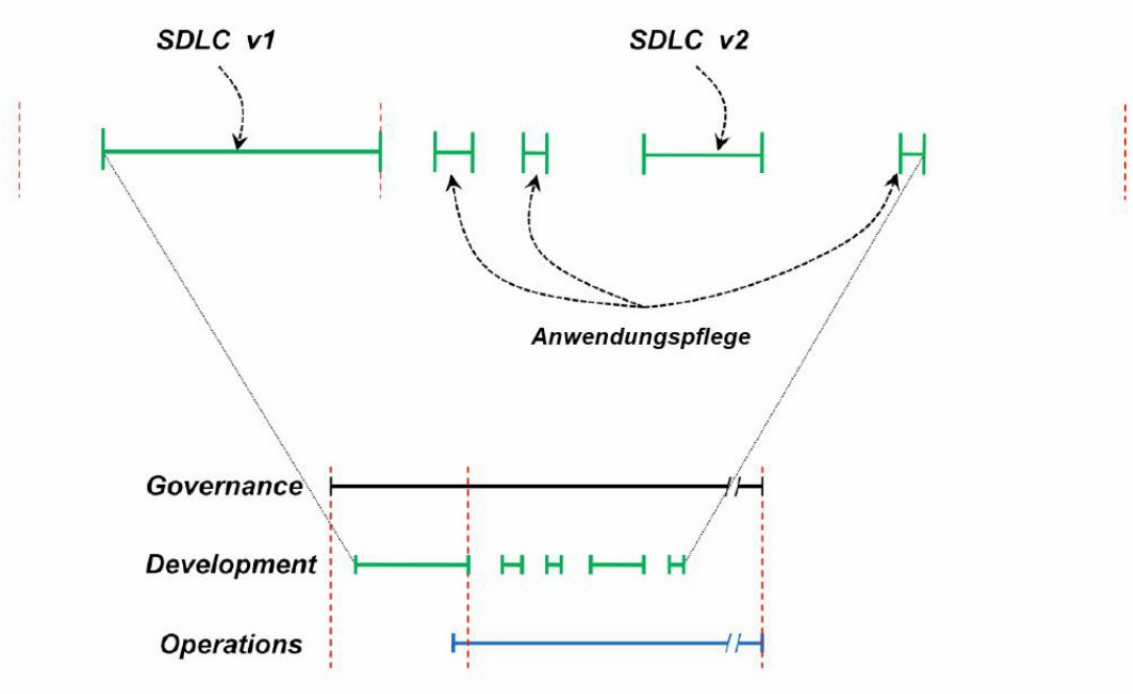


2.2 Development

- Grundlegender Teil des ALM
- Nach wirtschaftlicher Freigabe Start der Entwicklung für Version 1.0
  - Heutzutage oft iterativ nach agilen Methoden - Anforderungsbestimmung, Design, Implementierung/Test
  - Nach Abschluss erfolgt die Bereitstellung/Deployment der ersten Version
- Übergang in die Wartungs-/Maintenance-Phase
  - Bugfixes, Updates & Upgrades, …
  - Mitunter teurer als die initiale Entwicklung

2.2 Development

- Entwicklung neuer Versionen
  - z.T. heute rollende Updates & Upgrades, wie bei Chrome, Firefox, Windows 10, … Windows 11 ...

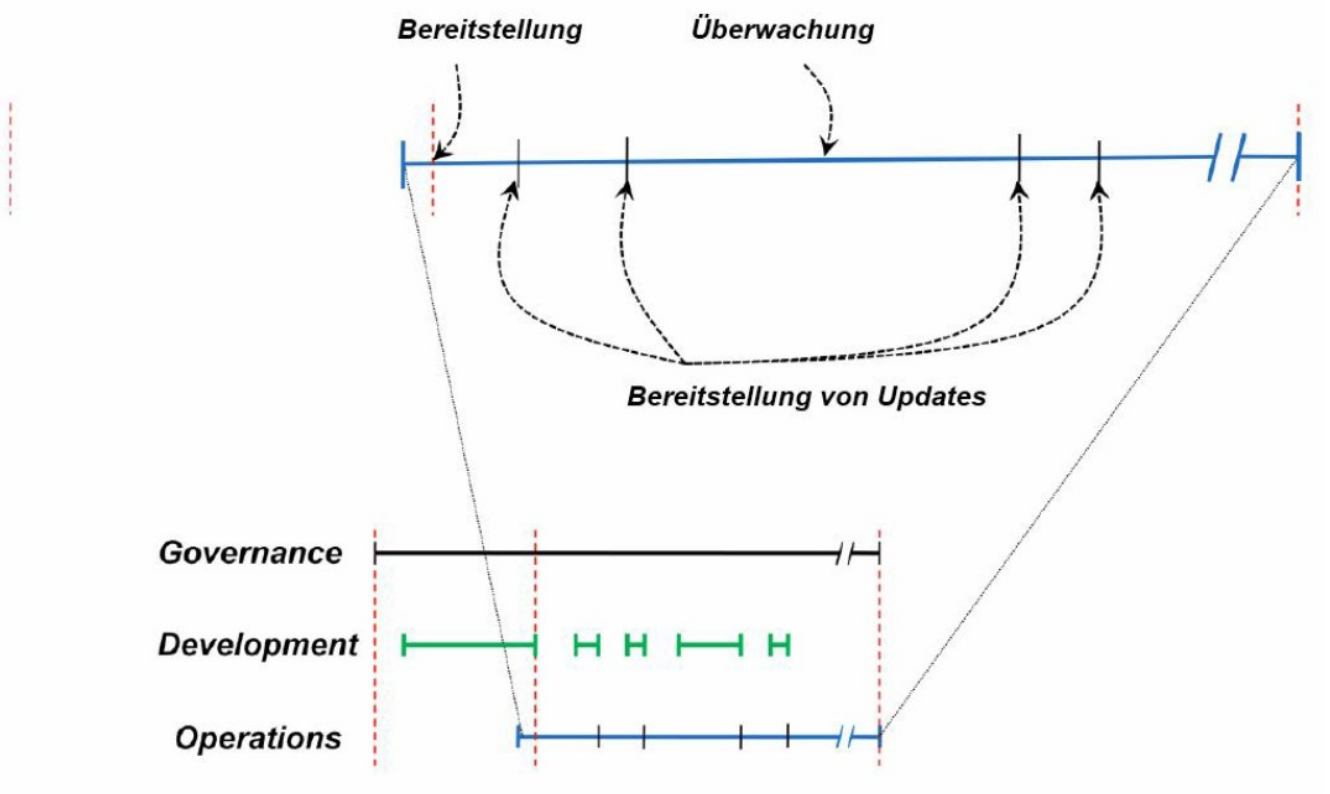


2.3 Operations

- Bereitstellung/Deployment nach der Freigabe zum „Ausrollen“ (bzw. eigentlich vorab schon die Vorbereitung dazu)
- Monitoring der Anwendung
  - Identifikation von Fehlerzuständen oder Engpässen, Skalierungsproblemen …
  - Direkte und schnelle Rückmeldung an die Entwicklung
- Update-Management
  - Technische Vorbereitung des Ausrollens von Updates und Upgrades

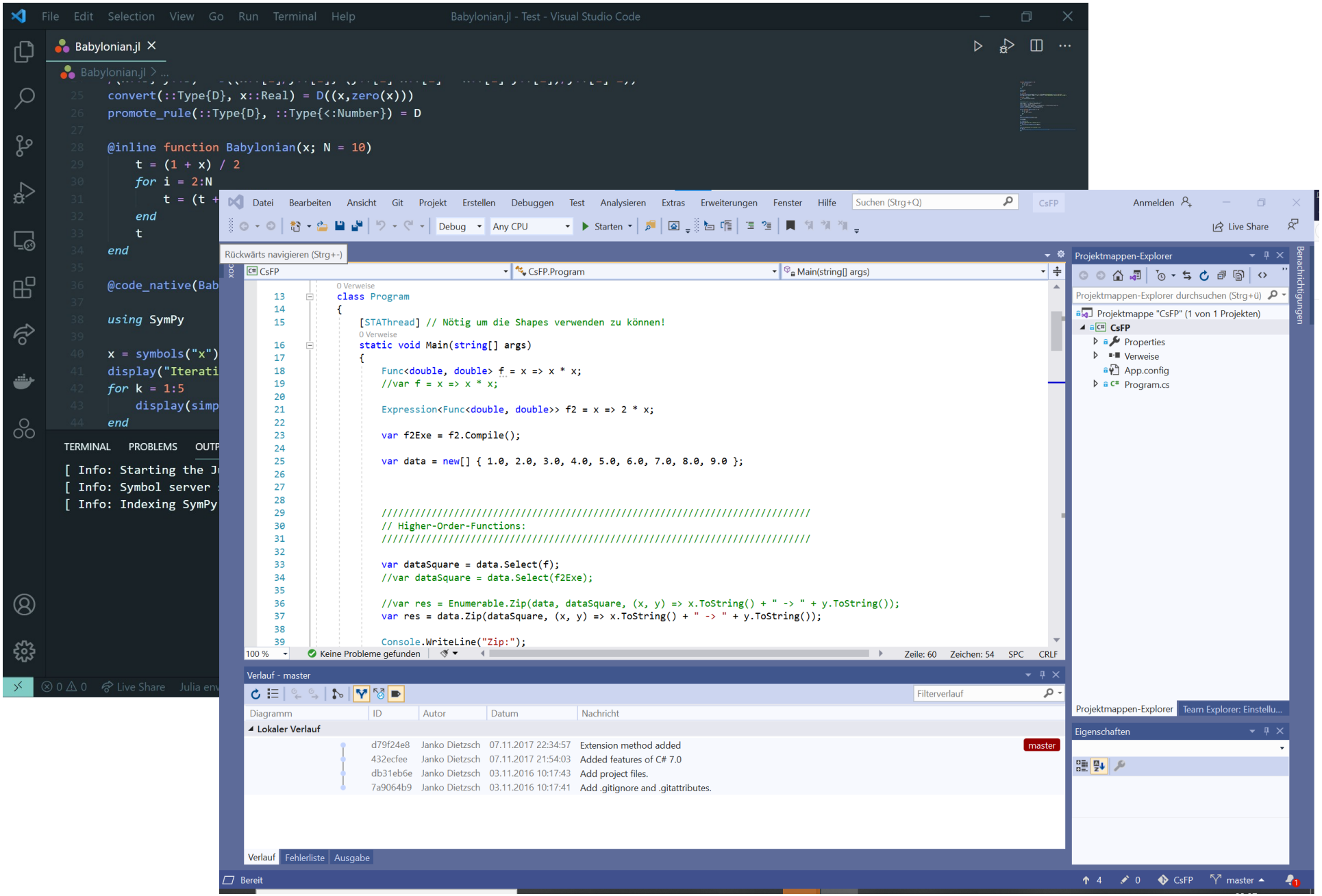
2.3 Operations

- Wächst immer enger mit der Entwicklung zusammen → DevOps



2.4 Werkzeuge für das ALM

- Integration der Tools nicht nur horizontal, sondern auch vertikal ...
  - Integration von IDE, Sourcecode Verwaltungssystem, ...
- Verbindung zw. Development & Operations
  - Virtualisierung
    - VM (VMware, VBox, ...)
    - Container (Docker, Podman, …)
    - Cloud
  - Infrastructure as Software (Software Defined Networks SDN, …)
    - Orchestrierung von VM's, Containern, … z.B. Kubernetes, Ansible, Puppet, Chef, Vagrant ...
  - Analytics (Health-Monitoring, ...)


File Edit Selection View Go Run Terminal Help
Babylonian.jl - Test - Visual Studio Code
Babylonian.jl
convert(::Type{D}, x::Real) = D((x,zero(x)))
promote_rule(::Type{D}, ::Type{<:Number}) = D
@inline function Babylonian(x; N = 10)
t = (1 + x) / 2
for i = 2:N
end
t
end
@code_native(Bab
using SymPy
x = symbols("x")
display("Iterati
for k = 1:5
display(simp
end
TERMINAL PROBLEMS
[Info: Starting the J
[Info: Symbol server
[Info: Indexing SymPy
Live Share
Datei Bearbeiten Ansicht Git Projekt Erstellen Debuggen Test Analysieren Extras Erweiterungen Fenster Hilfe
Suchen (Strg+Q)
CsFP
Anmelden
Debug
Any CPU
Starten
Live Share
Rückwärts navigieren (Strg+-)
CsFP
CsFP.Program
Main(string[] args)
0 Verweise
class Program
{
[STAThread] // Nötig um die Shapes verwenden zu können!
0 Verweise
static void Main(string[] args)
{
Func<double, double> f = x => x * x;
//var f = x => x * x;
Expression<Func<double, double>> f2 = x => 2 * x;
var f2Exe = f2.Compile();
var data = new[] { 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 };
//
// Higher-Order-Functions:
//
var dataSquare = data.Select(f);
//var dataSquare = data.Select(f2Exe);
//var res = Enumerable.Zip(data, dataSquare, (x, y) => x.ToString() + " -> " + y.ToString());
var res = data.Zip(dataSquare, (x, y) => x.ToString() + " -> " + y.ToString());
Console.WriteLine("Zip:");
100 %
Keine Probleme gefunden
Zeile: 60 Zeichen: 54 SPC CRLF
Projektmappen-Explorer
Projektmappen-Explorer durchsuchen (Strg+ü)
Projektmappe "CsFP" (1 von 1 Projekten)
CsFP
Properties
Verweise
App.config
Program.cs
Benachrichtigungen
Projektmappen-Explorer
Team Explorer: Einstellu...
Eigenschaften
Verlauf - master
Filterverlauf
Diagramm ID Autor Datum Nachricht
Lokaler Verlauf
d79f24e8 Janko Dietzsch 07.11.2017 22:34:57 Extension method added
master
432ecfee Janko Dietzsch 07.11.2017 21:54:03 Added features of C# 7.0
db31eb6e Janko Dietzsch 03.11.2016 10:17:43 Add project files.
7a9064b9 Janko Dietzsch 03.11.2016 10:17:41 Add .gitignore and .gitattributes.
Verlauf
Fehlerliste
Ausgabe
Bereit
4
0
CsFP
master

3. Versionsverwaltung

- Version Control System (VCS) – auch Source-Code Management (SCM) oder Revision-Control System (RCS)
- Aufgaben:
  - Genaue Protokollierung der Änderungen (Änderung, Zeitstempel, „ändernde Person“)
  - Archivierung jedes Versionsstandes eines Projektes
  - Wiederherstellung alter Versionsstände des Projektes
  - Ermöglicht die parallele Entwicklung an unterschiedlichen Zweigen/Branches
  - Kollaborationswerkzeug, koordiniert den Zugriff der Entwickler
- Archiv mit Historie, Wer hat wann was wie geändert?
- Einsatzorte:
  - Quellcodeverwaltung von Softwareprojekten
  - Content-Management Systemen
  - Office-Anwendungen (Change-Track-Mode)
  - Datenbanken (Audit-Trail)
  - ...

3.1 Begriffe

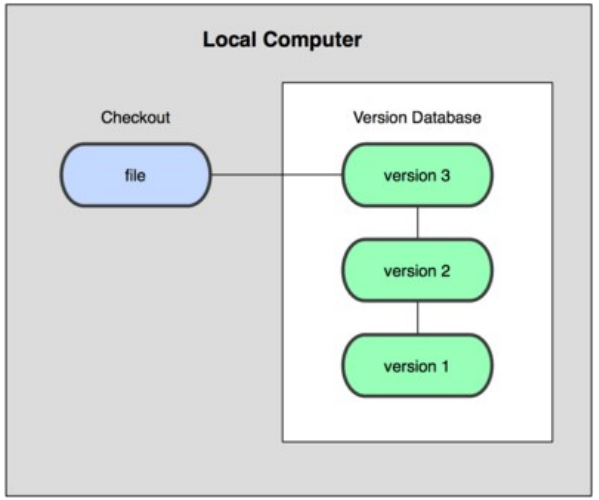
- ***Repository*** – Archiv bzw. Datenbank des VCS in dem die Historie abgelegt ist
- ***Working Directory/Arbeitskopie*** – aktuell in Bearbeitung befindlicher Stand des Projektes
- ***Checkout*** – synchronisieren der Arbeitkopie mit einem bestimmten Stand des Repositorys (aktuell oder älter)
- ***Fetch*** – Tracking-Informationen zum Remote-Repository aktualisieren
- ***Pull*** - Synchronisation des lokalen mit dem Remote-Repository
- ***Commit*** – synchronisieren des Repositorys mit dem aktuellen Zustand der Arbeitskopie
- ***Push*** - Synchronisation des lokalen mit dem Remote-Repository
- ***Branch/Zweig*** - erlaubt die gleichzeitige, parallele Entwicklung von verschiedenen Versionen
  - z.B. Trunk (SVN) oder Main (Git) für den Hauptzweig
- ***Merge*** – verschmelzen von Branches, z.B. ein Feature-Branch in den Master
- ***Tag*** – zusätzliche Annotationen um bestimmte Versionsstände zu markieren

3.2 Locking- bzw. Modifikationskonzepte

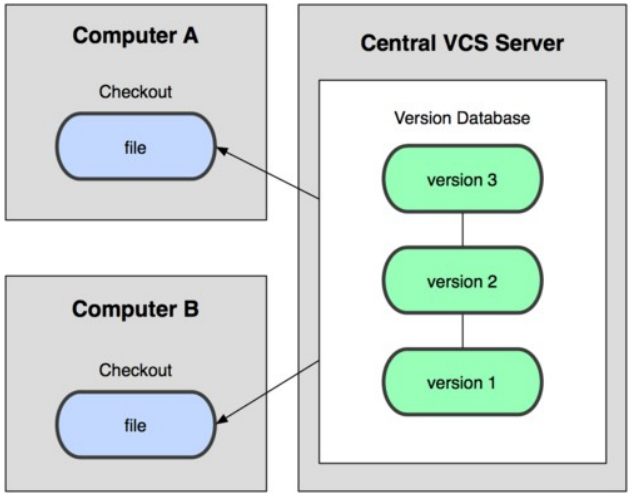
- Pessimistische Locking:
  - ***Lock Modify Write*** bzw. ***Lock Modify Unlock***: einzelne Dateien werden vor Änderung gesperrt und erst danach freigegeben (z.B. Visual SourceSafe)
  - Vorteil: keine Versionskonflikte, da die Dateien nicht zur gleichen Zeit bearbeitet werden kann
  - Nachteil: blockieren anderer Entwickler
  - Anmerkung: für Binärdateien oft unumgänglich
- Optimistisches Locking:
  - ***Copy Modify Write***: Kopieren der Datei → gleichzeitige Änderung → Zusammenführen der Änderungen hinterher

3.3 Verteilungskonzepte

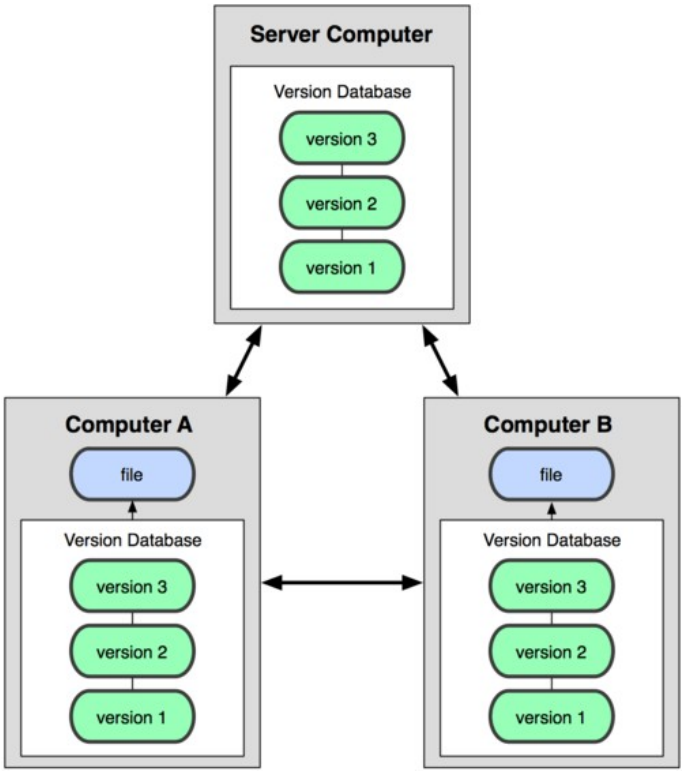
1. Lokal



2. Zentral



3. Verteilt



3.4 Historie

- Manuelles copy-basiertes „VCS“
- Open Source:
  - 1. Generation:

    SCCS (Source Code Control System), RCS (Revision Control System) ← pessimistische Locking & lokal
  - 2. Generation:

    Concurrent Version System (CVS) ← optimistischen Locking & zentral, Apache Subversion SVN, SVK mit lokaler Kopie des Repositorys
  - 3. Generation:

    Git, Mercurial, Bazaar, Darcs, Monotone … (verteilt)
- Kommerziell:
  - Visual SourceSafe (pessimistisches Locking & zentral)
  - Team Foundation Server, Perforce, ClearCase, BitKeeper

3.5 Git als modernes DVCS

https://xkcd.com/1597/

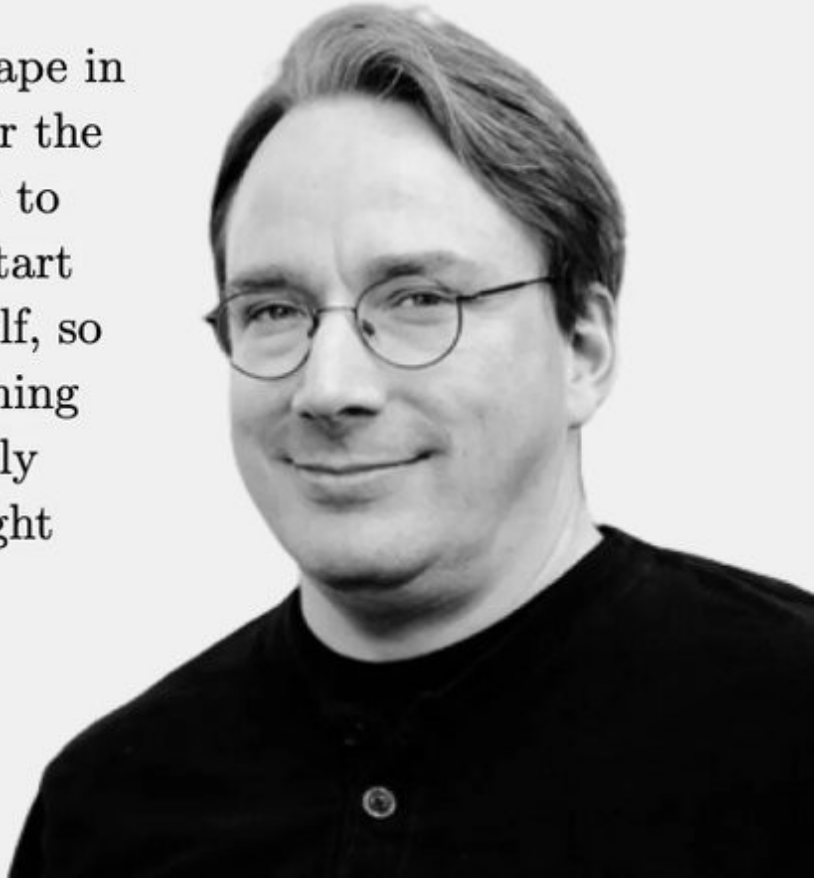
3.5 Git als modernes DVCS

- Wie kam es zu Git:
  - Entwicklung des Linux-Kernels von 1991 bis 2002 in Form von verschickten Archiven und Patches
  - Seit 2002 Nutzung des proprietären DVCS BitKeeper
  - ca. 2005 eskalierte die Diskussion um BitKeeper nachdem die freie Nutzung für Kernel-Entwickler zurückgezogen wurde
  - Linus Torvalds sah kein vorhandenes VCS als potentielle Alternative, SVN schon vorher abgelehnt, aufkommendes Python-basiertes Mercurial auch abgelehnt
  - Linus Torvalds startet die Entwicklung von Git als DVCS für die Kernel-Entwicklung → Shell-Scripte und C → anfänglicher schwerer Stand auf Windows als Mercurial
    - Ziele: verteilt, robust, effizient
  - Seit 2005 auf allen Plattformen gut entwickelt und ausgereift
  - Sehr weit verbreitet, gute Unterstützung durch Hoster (Github, Bitbucket, GitLab, …)
  - Steilere Lernkurve als beispielsweise Mercurial, aber sehr viel Möglichkeiten

3.5 Git als modernes DVCS

"You can actually see how it all took shape in the git source code repository, except for the very first day or so. It took about a day to get to be "self-hosting" so that I could start committing things into git using git itself, so the first day or so is hidden, but everything else is there. The work was clearly mostly during the day, but there's a few midnight entries and a couple of 2am ones"

- Linus Torvalds



Quellen

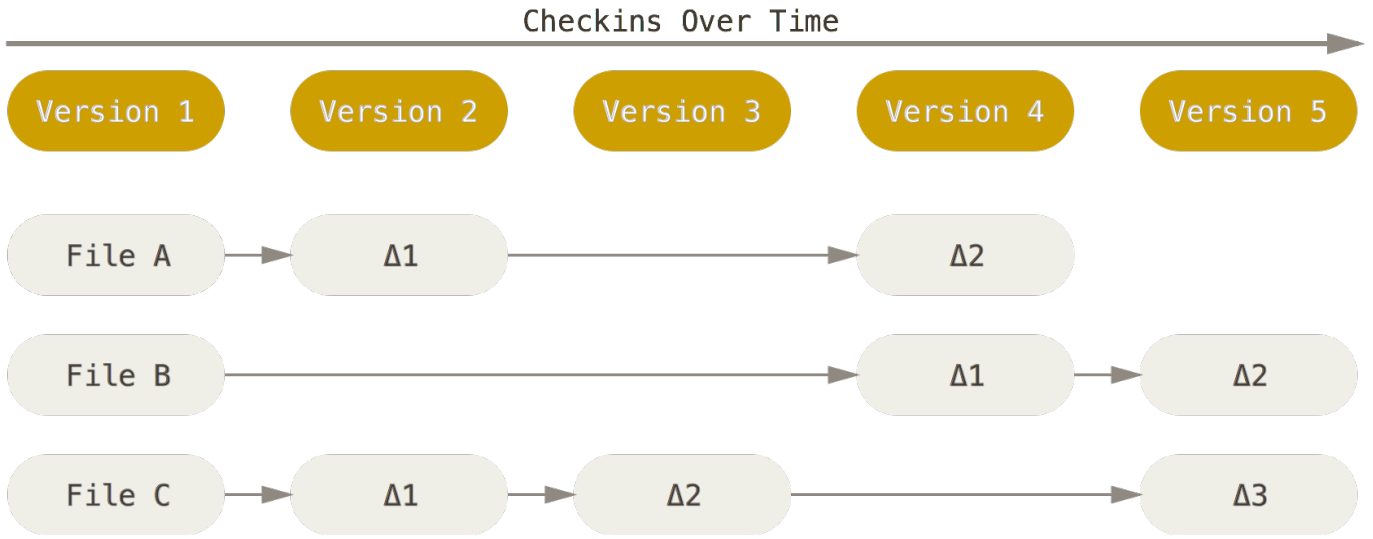
- „Pro Git“, 2ed Edition (2014) von Scott Chacon und Ben Straub, https://www.git-scm.com/book/en/v2
  - Kann auch als Ebook runtergeladen werden
  - 1.Auflage auch in Deutsch verfügbar - https://git-scm.com/book/de/v1
  - Eine ganze Reihe von hier verwendeten Abbildungen sind aus diesem Buch
- Getting Git Right (Atlassian) https://www.atlassian.com/git/
- ...

Anmerkung

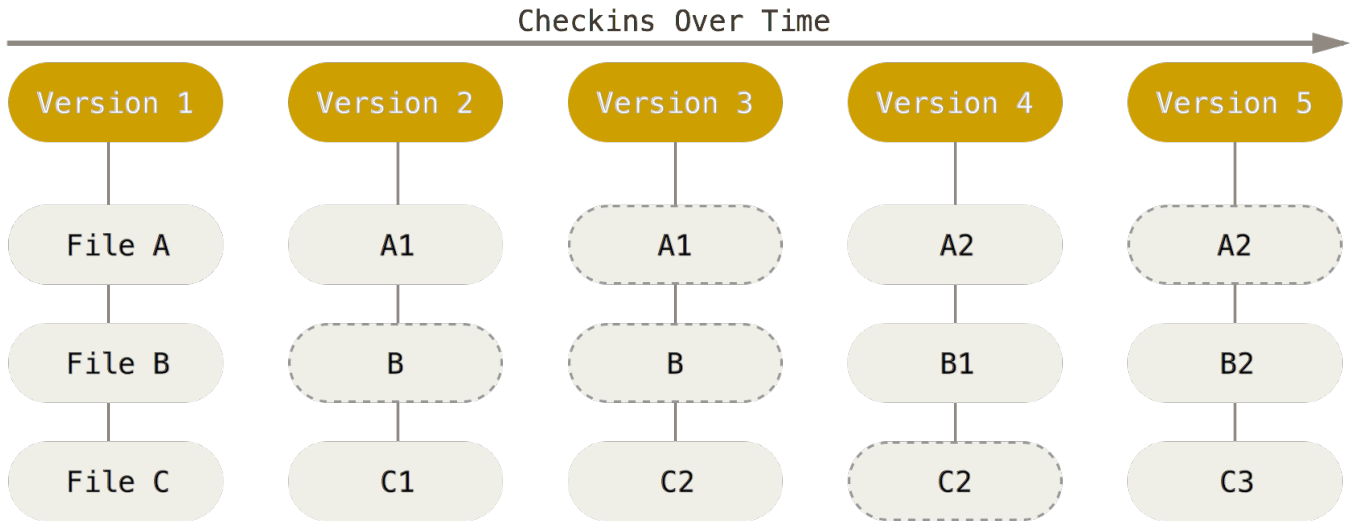
- Aufgrund von Diskussionen um diskriminierende Sprache bzw. diskriminierende Sprachkonstrukte, erfolgte eine Empfehlung zur Umbenennung des Default-Branches von ***master*** in ***main***
- GitHub begann ab dem 1.10.2020 mit der Umstellung des Defaults für neu angelegte Repositories
- In der Vorlesung wird noch der alte Default ***master*** verwendet

SVN vs Git – Deltas vs. Snapshots

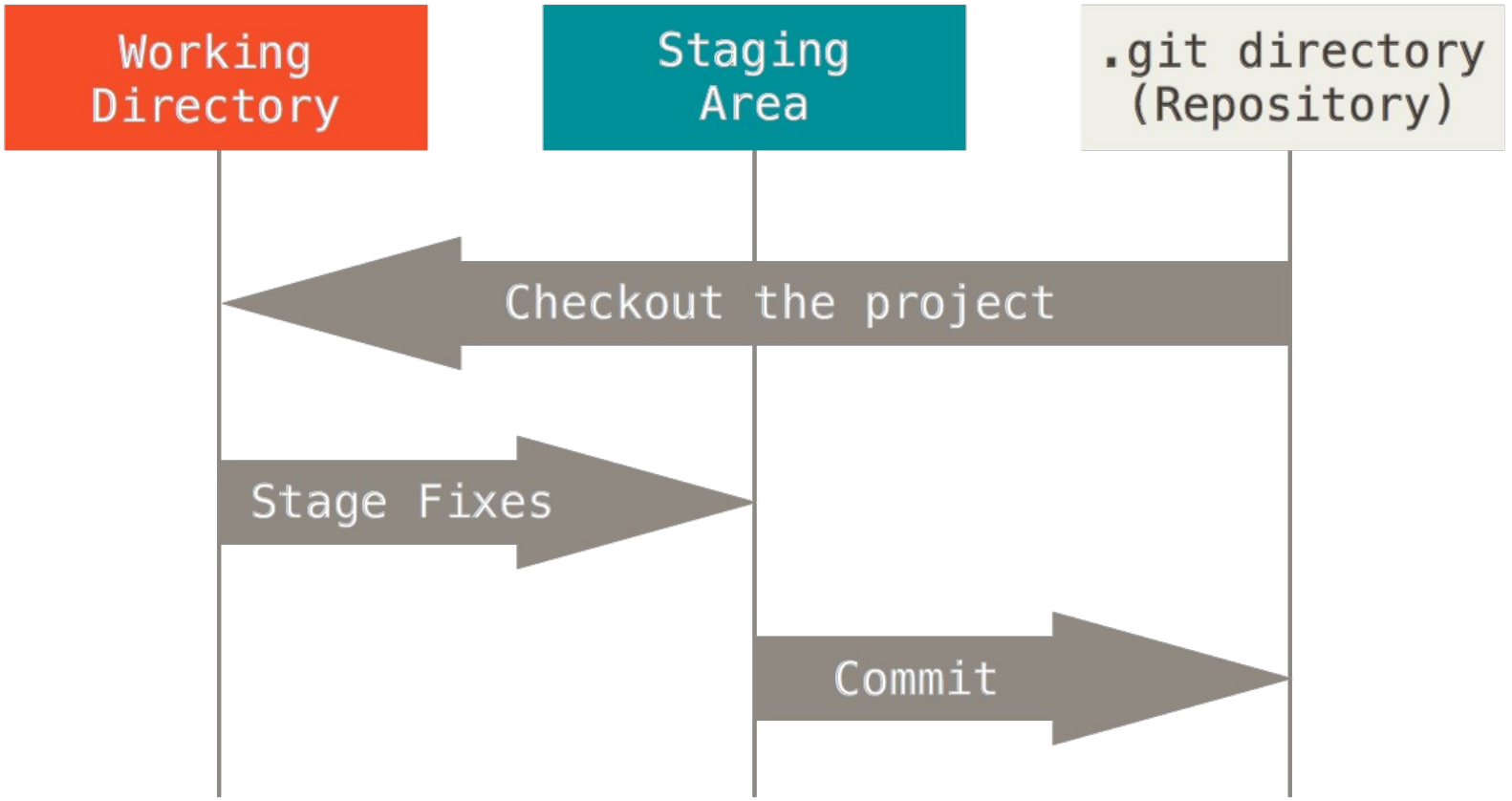
Deltas:



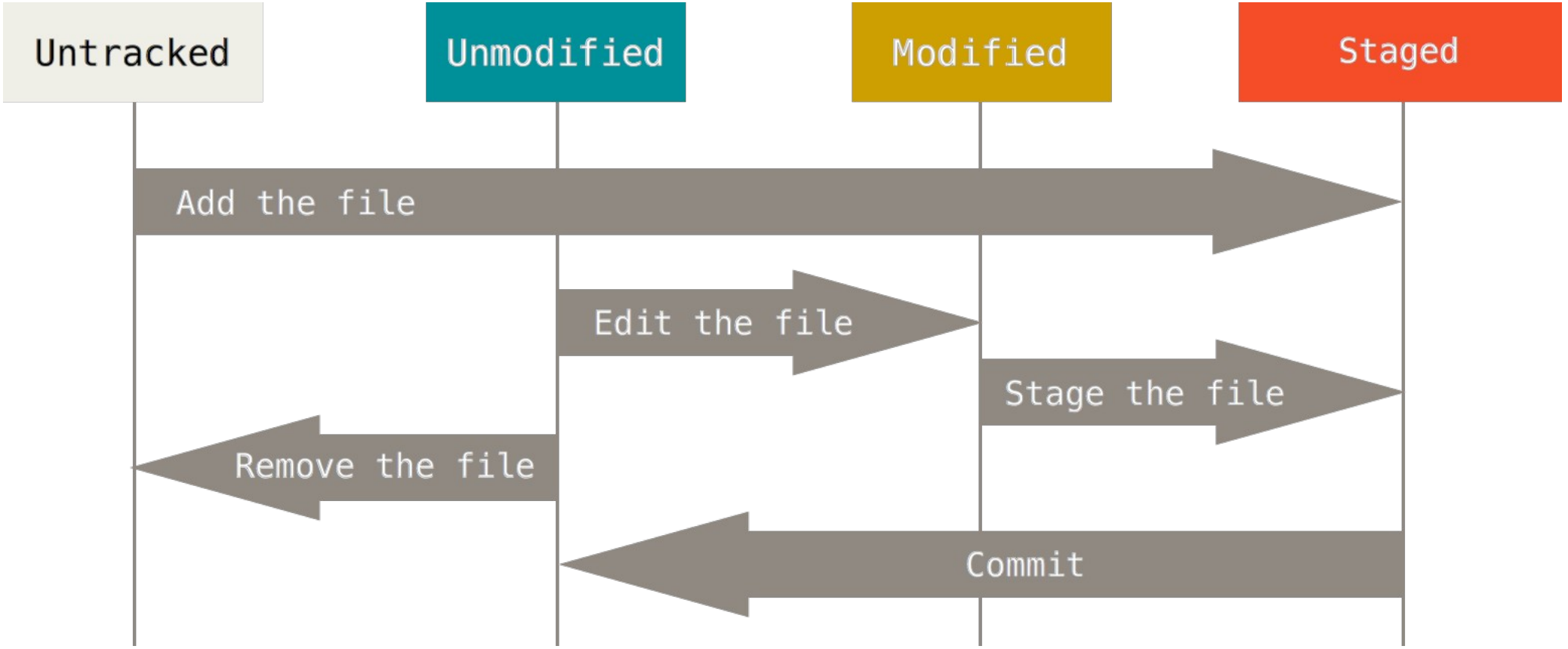
Snapshots:



Die drei Stufen von Git



Zustände von Dateien in Bezug auf das Repository



Git im Inneren

- **.git** -Verzeichnis

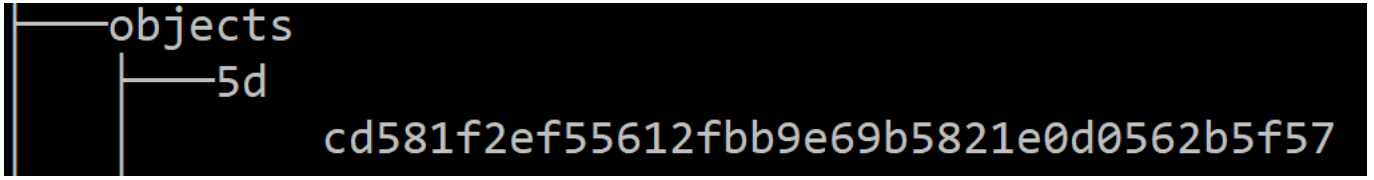
initialisiert

```
config
description
HEAD

hooks
    applypatch-msg.sample
    commit-msg.sample
    post-update.sample
    pre-applypatch.sample
    pre-commit.sample
    pre-push.sample
    pre-rebase.sample
    prepare-commit-msg.sample
    update.sample

info
    exclude

objects
    info
    pack
refs
    heads
    tags
```

git add test.txt

```
config
description
HEAD
index

hooks
    applypatch-msg.sample
    commit-msg.sample
    post-update.sample
    pre-applypatch.sample
    pre-commit.sample
    pre-push.sample
    pre-rebase.sample
    prepare-commit-msg.sample
    update.sample

info
    exclude

objects
    5d
        cd581f2ef55612fbb9e69b5821e0d0562b5f57

    info
    pack
refs
    heads
    tags
```

Git im Inneren



- Datei *Test.txt* wird SHA1 gehasht:
  - SHA1(**Content**) → 40 Ziffern → **Key**
  - **Content** → Blob (Binary Large Object) → **Value**
- **.git/objects**
  - Erste zwei Ziffern des SHA1 → Verzeichnis-Name
    - Weitere 38 Ziffern als Dateiname des Blobs
- Key/SHA1 → Value/Blob
- *Inhaltsadressierbarer Key-Value-Store*

Git im Inneren

Nach einiger Zeit ...

```
$ find .git/objects -type f
.git/objects/00/30da6c95295ed0b4bb40e3973b97def744ed10
.git/objects/03/bc7252dcf0d2f6969434299aa3052ae28245c3
.git/objects/06/4a28ddb458715659edfa3abaa74b499277ef62
.git/objects/06/815ae304eff9dceaf726e8797607a8c4ff1753
.git/objects/09/11a924c477c41e05926180cf988482e9a0daa2
.git/objects/0b/eb7f557ebc36f0820d28e22d30f0d974c922b7
.git/objects/0c/71d201edd85696f2a16c2b23f53e25defbc275
.git/objects/0d/f5a7797f847472b056728383eb94ac361138ce
.git/objects/0e/90923bd7525fb37ddb16ee843d257051511c23
.git/objects/0f/40676641db3d429546ce7aa64368484af39f60
.git/objects/0f/5cffd82a065e09f68bdc7943a60fec2be3b8b0
.git/objects/10/30bcf7d013fba55653c40086e67ae0f2b72566
.git/objects/14/a3d41a8b2966ef0cc33c3cc0505f536f95f92a
.git/objects/16/4145c354af6f908b0fe9df88e6425be320c6df
.git/objects/16/7a17b76c723a2524632ca5620c746f41c55d5c
.git/objects/16/9fa2e084b71888b35afbe1285b763a4a0f1042
```

Git im Inneren

```
C:\Users\Janko\Documents\Projekte\GitStuff\Test>git commit -m "Initial Commit"
[master (root-commit) a6bd3c6] Initial Commit
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

C:\Users\Janko\Documents\Projekte\GitStuff\Test>tree /F .git\objects
Auflistung der Ordnerpfade für Volume Windows
Volumeseriennummer : A0C5-1031
C:\USERS\JANKO\DOCUMENTS\PROJEKTE\GITSTUFF\TEST\.GIT\OBJECTS
├───5d
│       cd581f2ef55612fbb9e69b5821e0d0562b5f57
│
├───87
│       1f3e3ac97cc09d42031cebe7bf0dc963d47273
│
├───a6
│       bd3c65a973b004d44c7d1c57e255b9bbc56630
│
├───info
└───pack
```

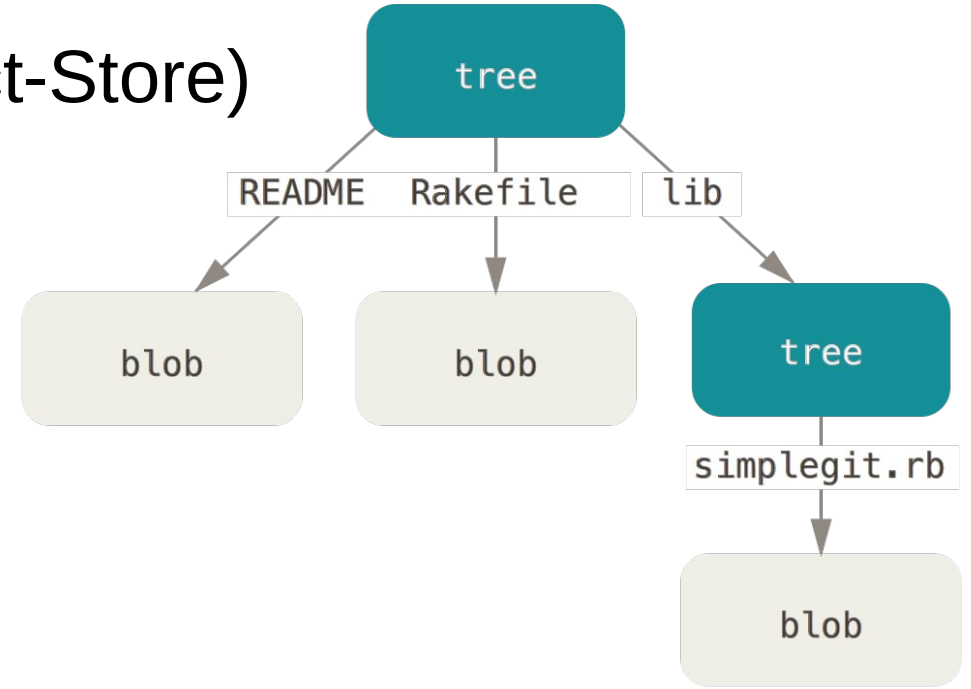
Git im Inneren

```
Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$ git cat-file -t 5dcd581f2ef55612fbb9e69b5821e0d0562b5f57
blob

Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$ git cat-file -t 871f3e3ac97cc09d42031cebe7bf0dc963d47273
tree

Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$ git cat-file -t a6bd3c65a973b004d44c7d1c57e255b9bbc56630
commit
```

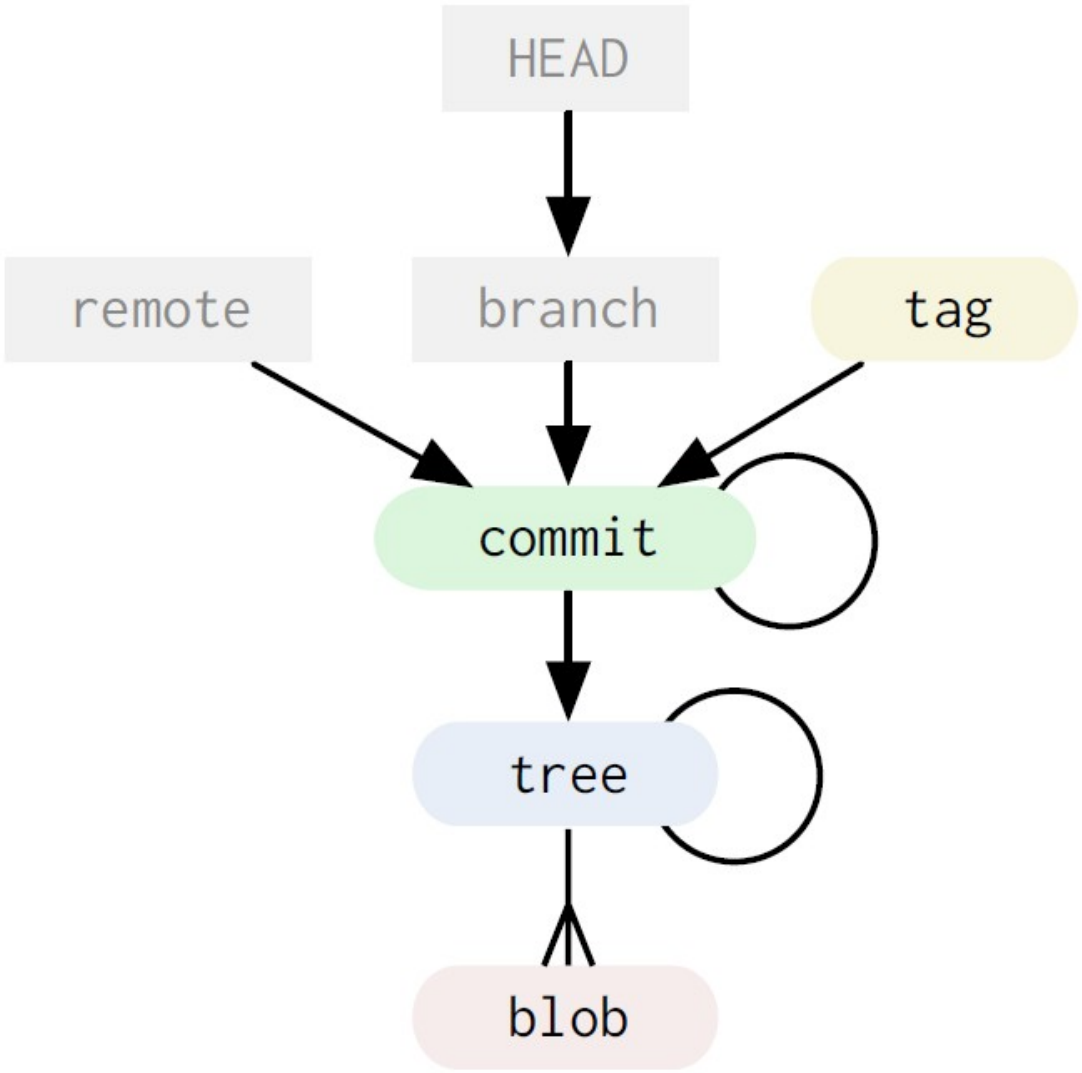
```
Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$ git cat-file -p 5dcd581f2ef55612fbb9e69b5821e0d0562b5f57
bla
Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$ git cat-file -p 871f3e3ac97cc09d42031cebe7bf0dc963d47273
100644 blob 5dcd581f2ef55612fbb9e69b5821e0d0562b5f57    test.txt

Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$ git cat-file -p a6bd3c65a973b004d44c7d1c57e255b9bbc56630
tree 871f3e3ac97cc09d42031cebe7bf0dc963d47273
author Janko Dietzsch <JankoDietzsch@web.de> 1522961279 +0200
committer Janko Dietzsch <JankoDietzsch@web.de> 1522961279 +0200

Initial Commit

Janko@athene MINGW64 ~/Documents/Projekte/GitStuff/Test (master)
$
```

Git im Inneren

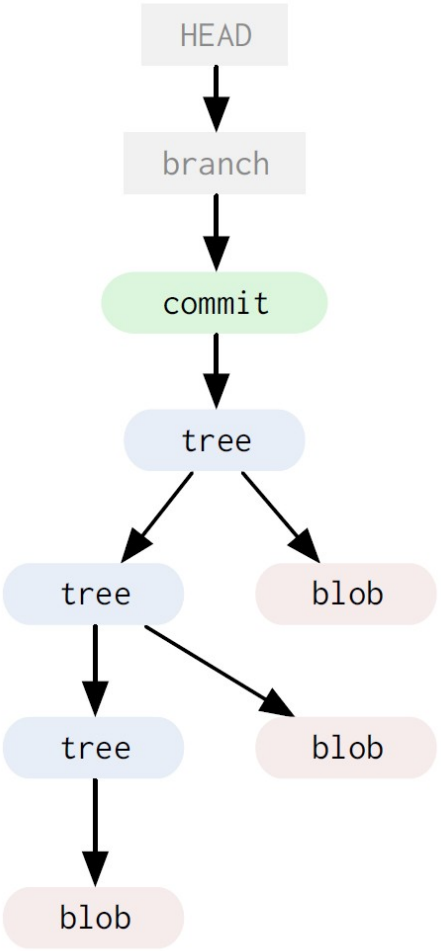
- Trees, Blobs & Commits
  - „Inhaltsaddressierbares Filesystem“
  - Hashes als Keys
  - Key-Value-Store (Object-Store)



Git im Inneren - „Datenmodell“

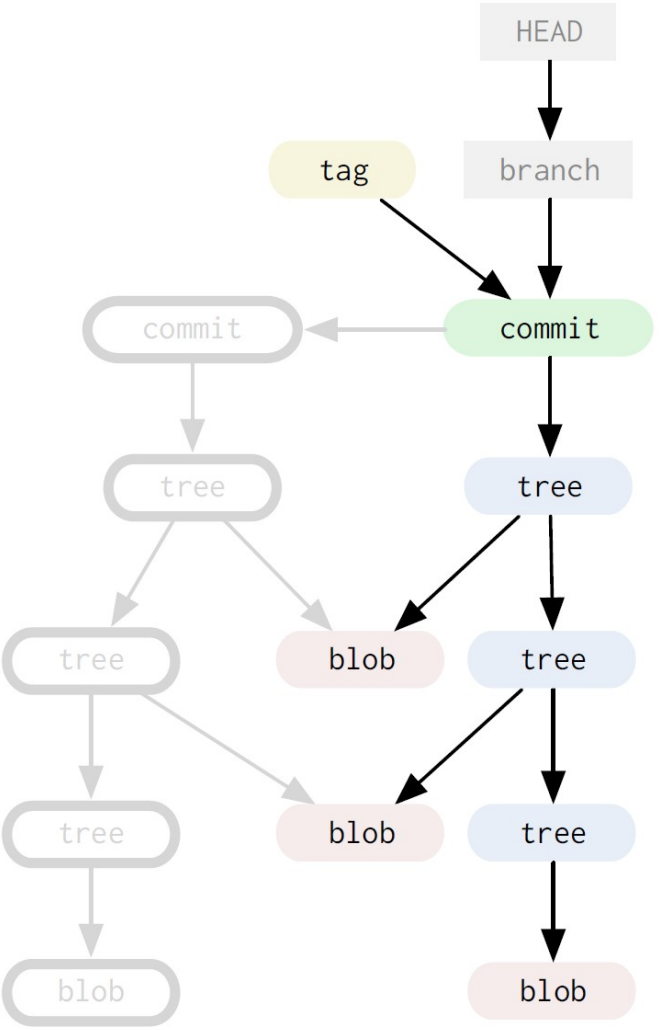


Git im Inneren



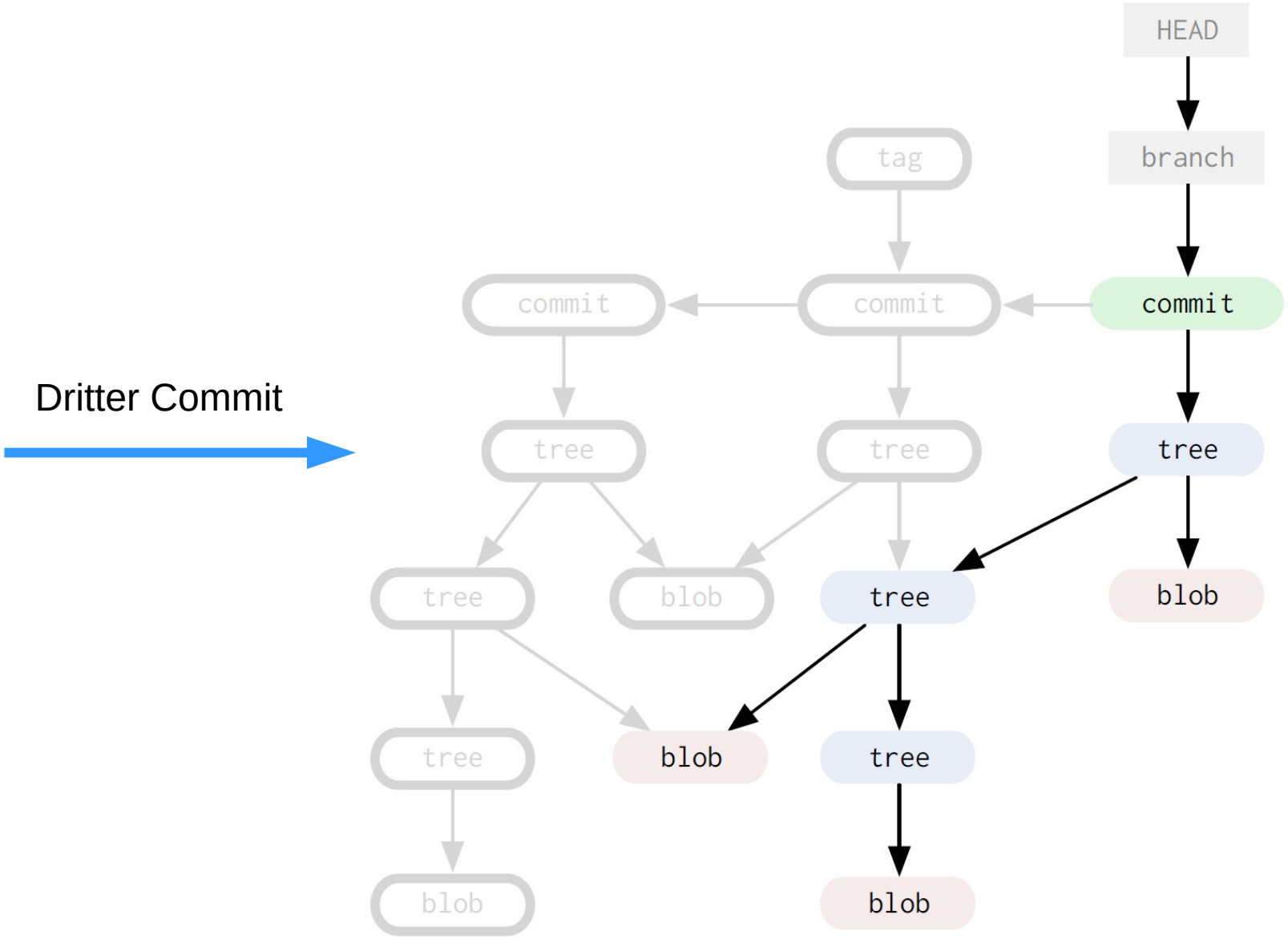
Erster Commit

Zweiter Commit



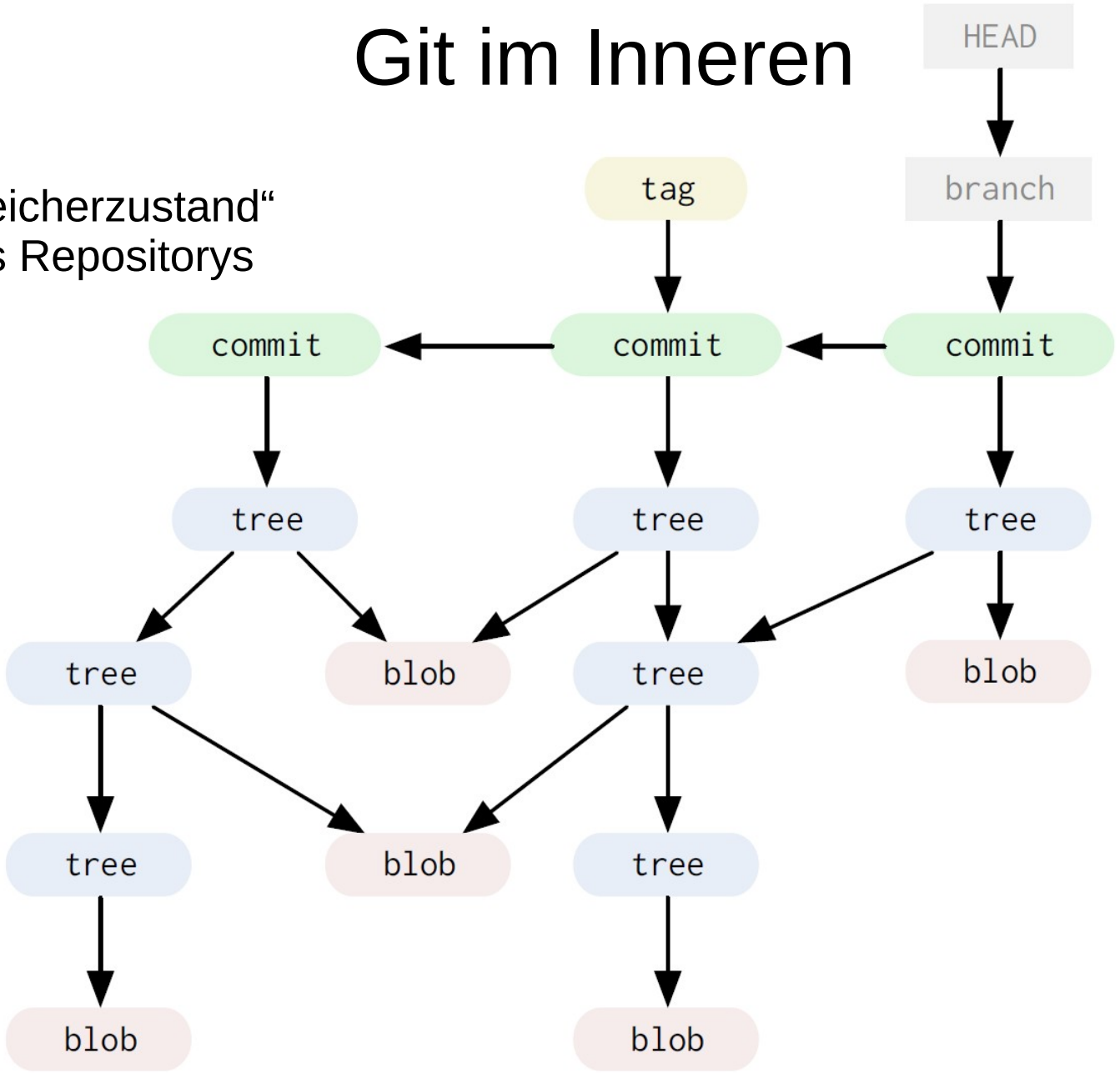
Git im Inneren

Dritter Commit



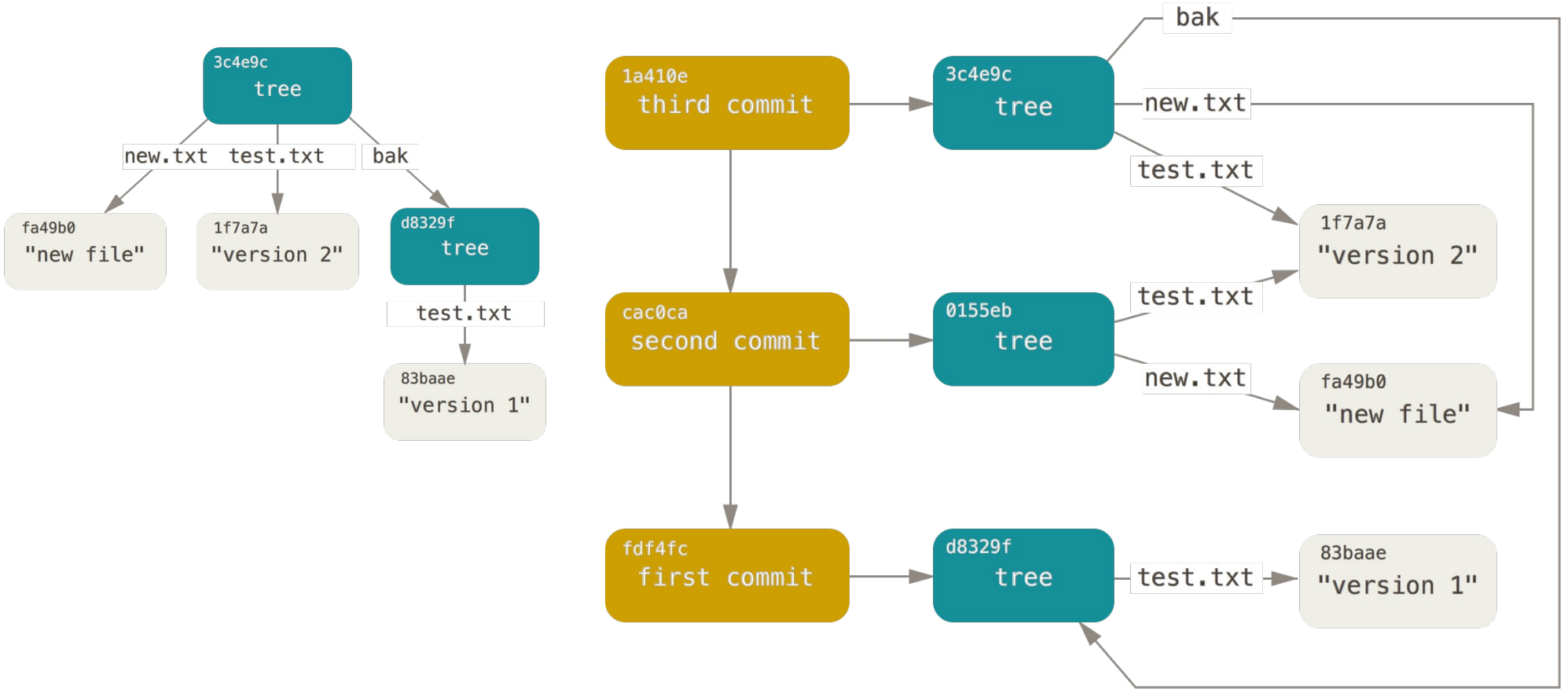
Git im Inneren

„Speicherzustand“
des Repositorys



Git im Inneren

- Historie von Commits/Snapshots:



Git im Inneren

- Dinge die mit Git Probleme bereiten können:
  - Große binäre Dateien → LFS als Lösung
  - Sehr große Repositories, die z.B. aus mehreren Projekten bestehen → verlagern in eigene Repositories & Abhängigkeiten durch Package-Manager der Sprache lösen (z.B. NuGet, Bundler, ...)
  - Problematik der Subprojekte (Submodules, Subtree, Subrepo…)
  - „Monorepo“ vs. „Multi-Repo“

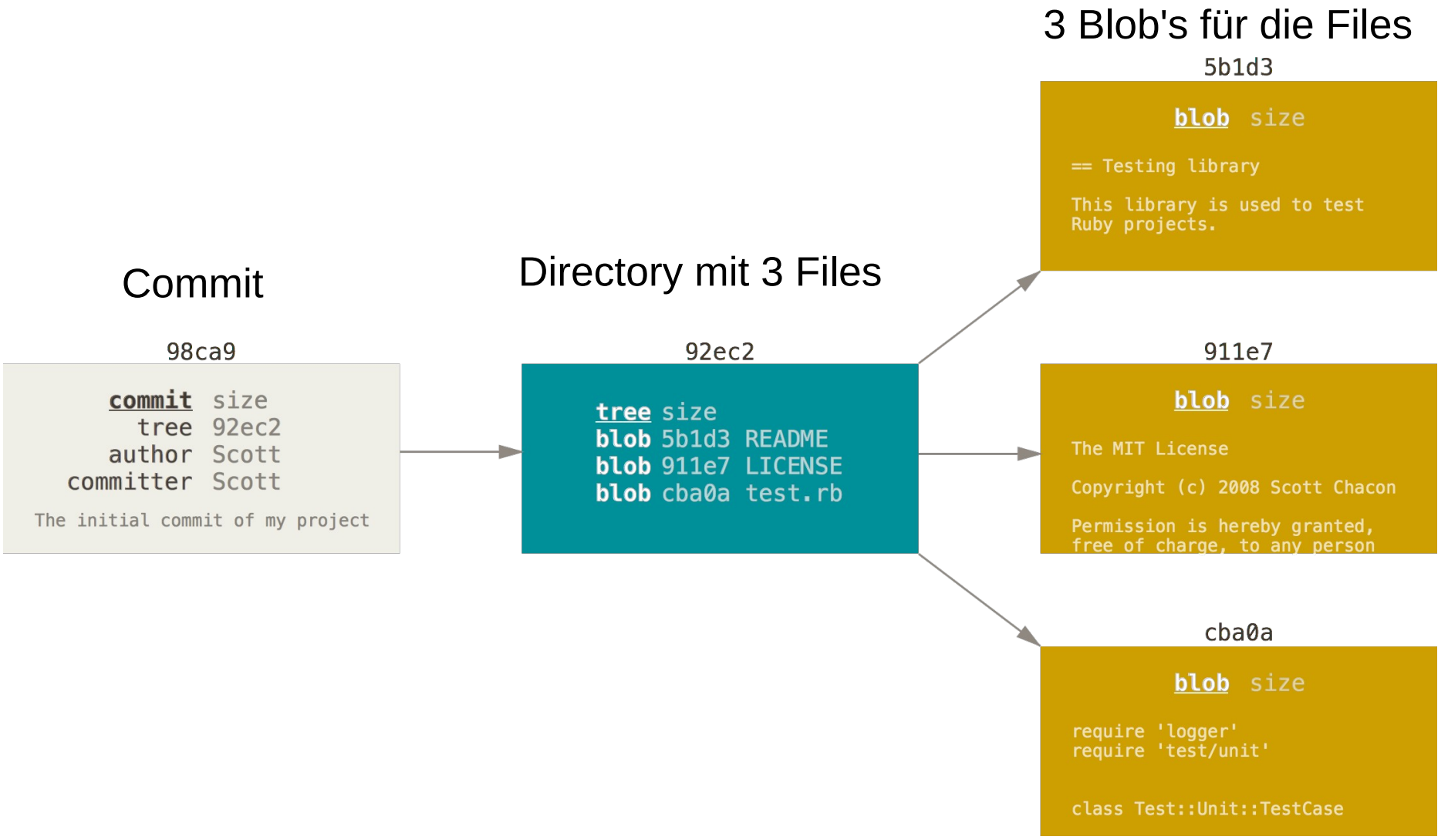
Windows Code-Basis als Beispiel

- Umfangreiche Dimensionen:
  - 3.5 Millionen Files
  - 270 GB Größe
- Konsequenzen:
  - *git checkout* → bis zu 3 h
  - *git status* → bis zu 12 min
  - *git clone* → mehr als 12 h
- Vorschlag eines GVFS (Git Virtual File System)

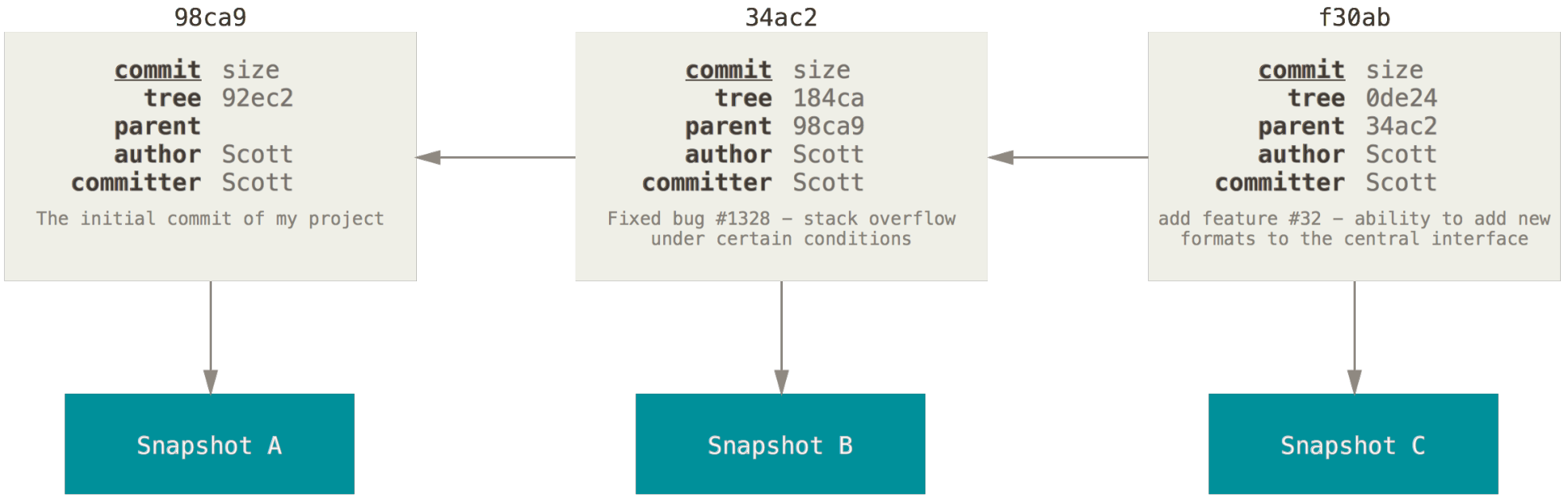
  (https://github.com/Microsoft/gvfs), dass das Filesystem unterhalb des Git-Repositorys virtualisieren soll

https://blogs.msdn.microsoft.com/visualstudioalm/2017/02/03/announcing-gvfs-git-virtual-file-system/

Von Commits zu ...

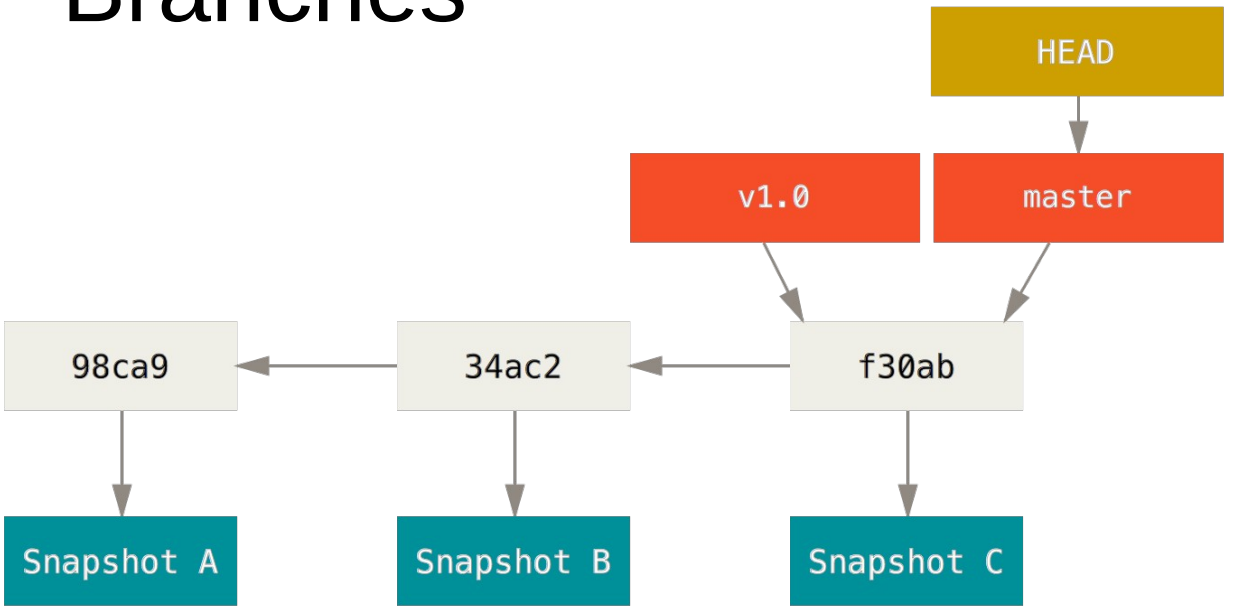


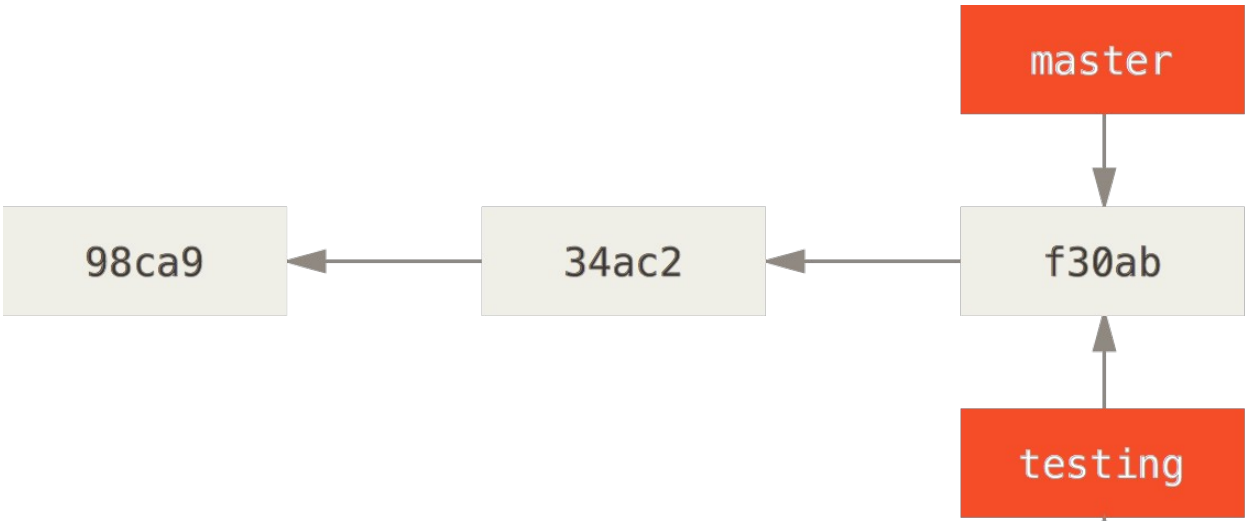
Von Commits zu … Branches



Branches

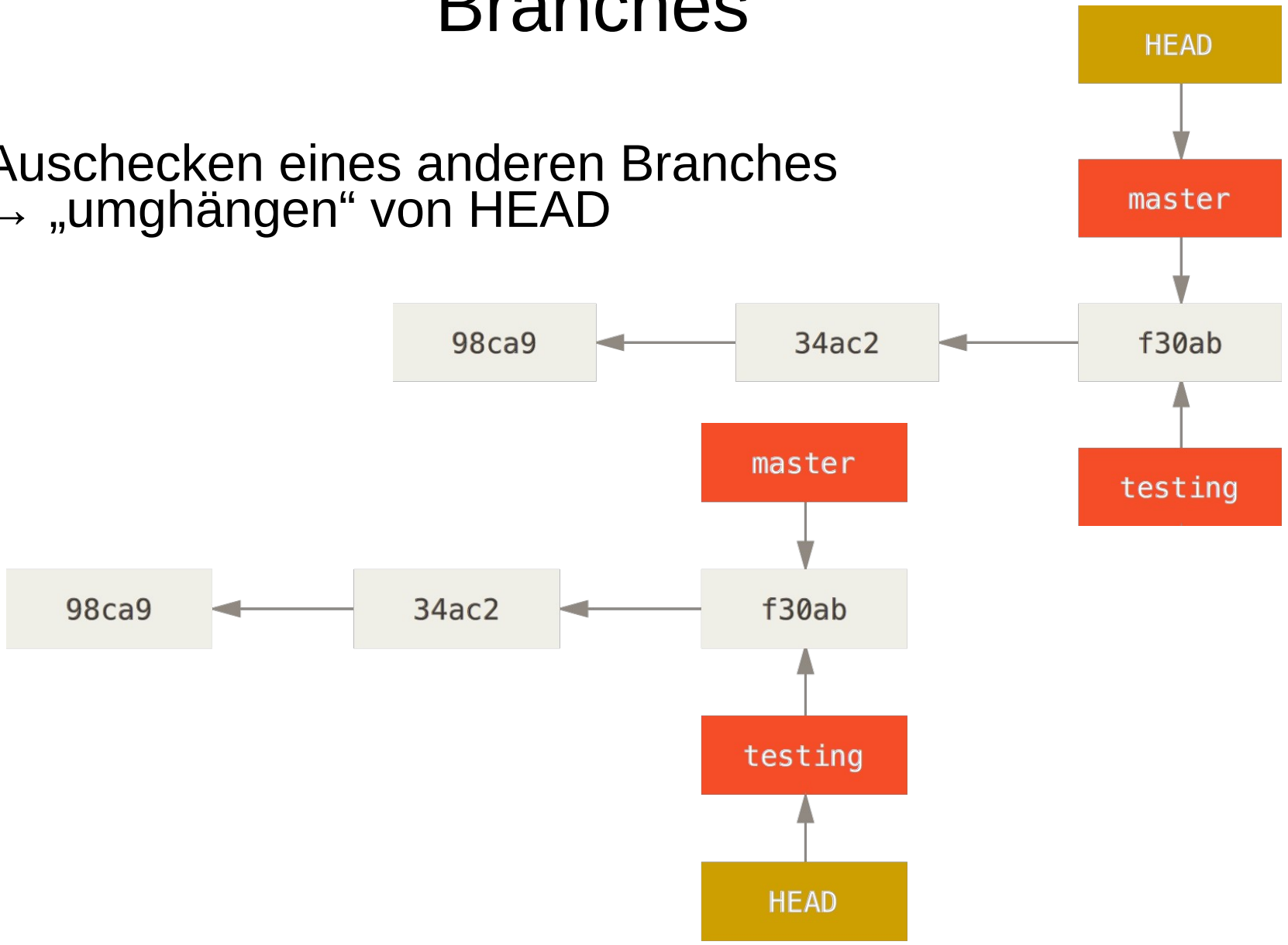
- Branches sind lediglich Pointer
- HEAD zeigt auf den aktuell ausgecheckten Branch





Branches

- Auschecken eines anderen Branches
  → „umghängen“ von HEAD



Branches

- Neuer Commit → HEAD bewegt sich mit
- Auschecken von ***master***