

An open architecture for AI-native development

Ty Dunn, Co-founder of  Continue

Goals for this talk

1. Describe one of many possible open architectures for AI-native development
2. Inspire you to think more deeply about the architecture underlying the future of software development

What is Continue?

Open-source IDE extensions

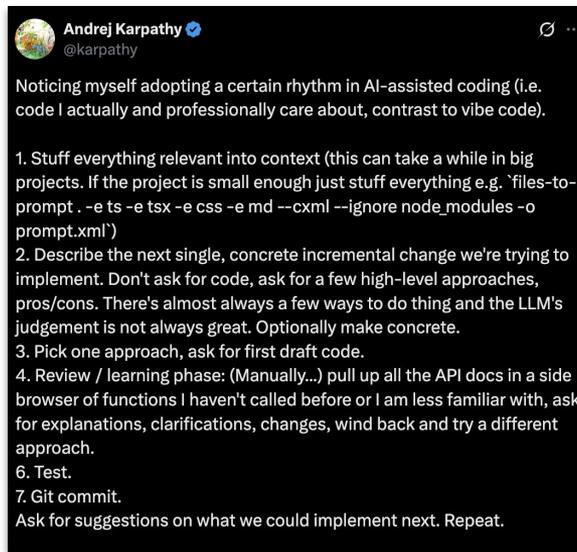
Action	Target	Description
Client component	APIRoute	Create a client component
Create a client component with the fol	Create an API route with the following	Create a new Next.js page based on th...
Prisma schema	Server component	Create a Prisma schema
Create or update a Prisma schema with	Create a server component with the fol	
 The right sidebar shows a 'Models' list with 'Claude 3.7 Sonnet' selected.

Hub of rules, tools, and models to create, share, and use custom AI code assistants

What is AI-native development?

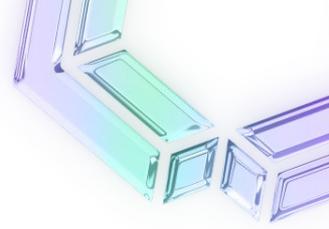


Vibe coding

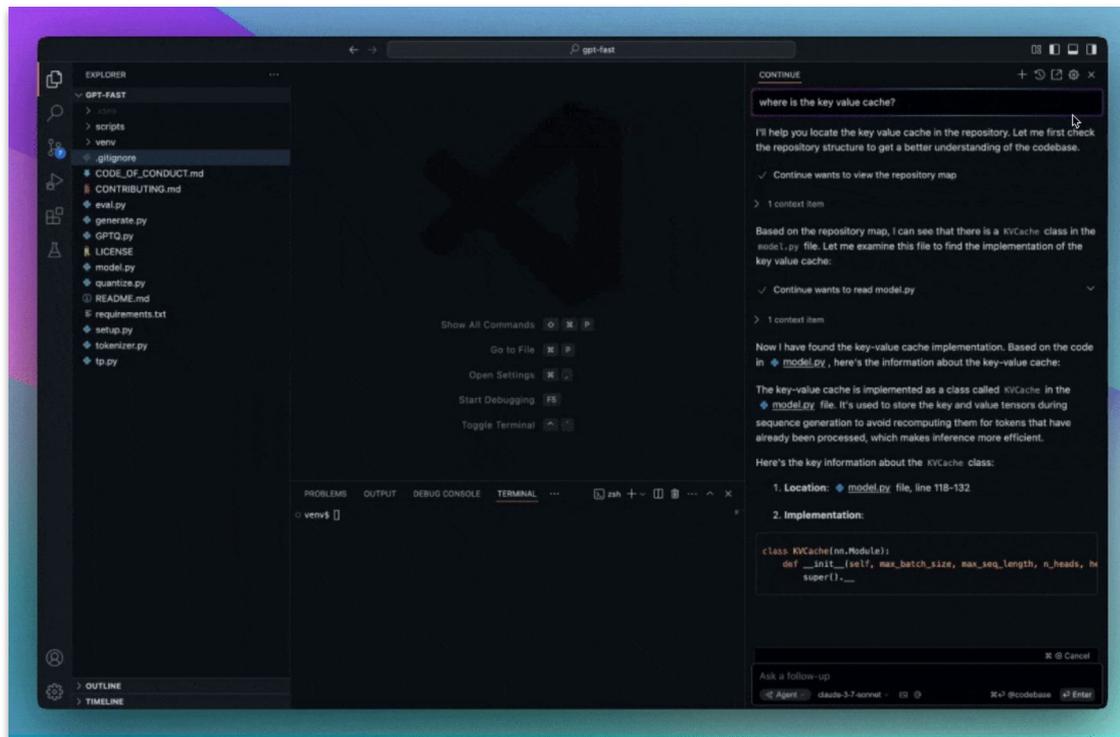


AI-native development

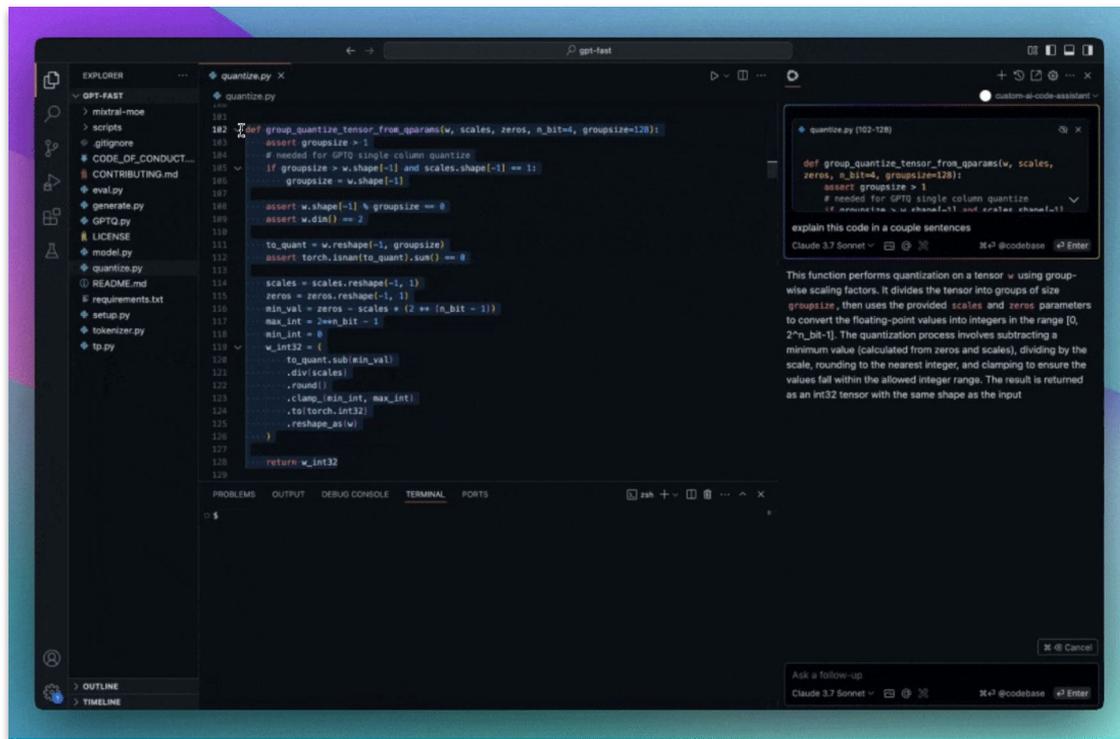
An example AI-native development workflow



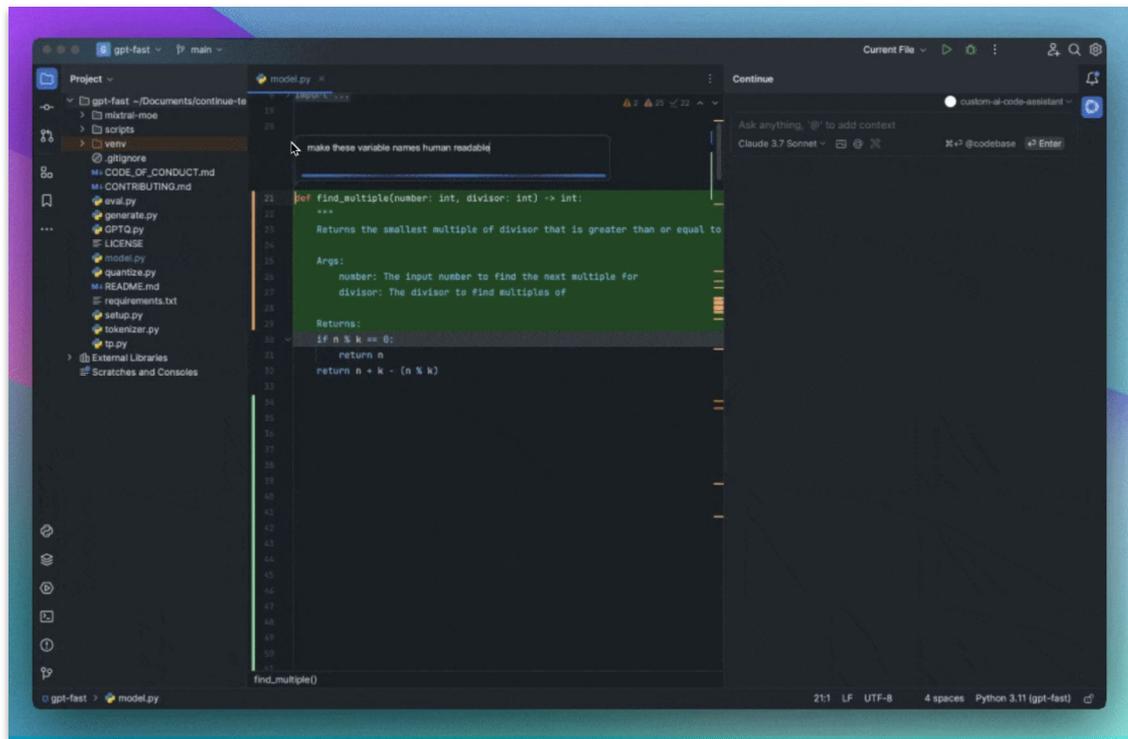
Agent to ask a model to do a task for you



Chat to understand and iterate on code with a model



Edit to transform a section of code using a model



Autocomplete to have a model finish lines of code

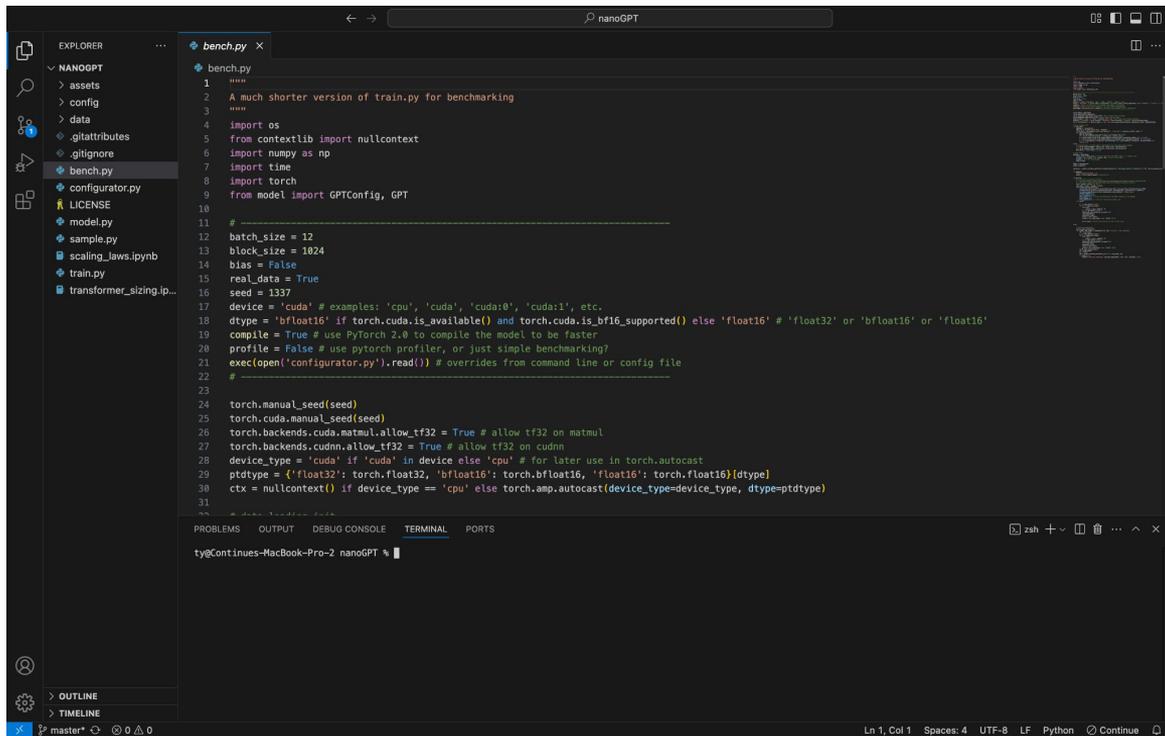
```
ll.py

if __name__ == '__main__':
    import argparse
    parser = argparse.ArgumentParser(description='Your CLI description.')

    parser.add_argument('--checkpoint_path', type=Path, default=Path("checkpoints/meta-llama/Llama-2-7b"))
    parser.add_argument('--compile', action='store_true', help='Whether to compile the model.')
    parser.add_argument('--tasks', nargs='+', type=str, default=["hellaswag"], help='list of lm-eval-harness tasks')
    parser.add_argument('--limit', type=int, default=None, help='number of samples to evaluate')
    parser.add_argument('--max_seq_length', type=int, default=2048, help='max sequence length')

    args = parser.parse_args()
    main(
        Path(args.checkpoint_path), args.compile, args.tasks, args.limit, args.max_seq_length,
    )
```

Write the code manually without using a model



```
1 """
2 A much shorter version of train.py for benchmarking
3 """
4 import os
5 from contextlib import nullcontext
6 import numpy as np
7 import time
8 import torch
9 from model import GPTConfig, GPT
10
11 # -----
12 batch_size = 12
13 block_size = 1024
14 bias = False
15 real_data = True
16 seed = 1337
17 device = 'cuda' # examples: 'cpu', 'cuda', 'cuda:0', 'cuda:1', etc.
18 dtype = 'bfloat16' if torch.cuda.is_available() and torch.cuda.is_bfloat16_supported() else 'float16' # 'float32' or 'bfloat16' or 'float16'
19 compile = True # use PyTorch 2.0 to compile the model to be faster
20 profile = False # use pytorch profiler, or just simple benchmarking?
21 exec(open('configurator.py').read()) # overrides from command line or config file
22 # -----
23
24 torch.manual_seed(seed)
25 torch.cuda.manual_seed(seed)
26 torch.backends.cudnn.allow_tf32 = True # allow tf32 on matmul
27 torch.backends.cudnn.allow_tf32 = True # allow tf32 on cudnn
28 device_type = 'cuda' if 'cuda' in device else 'cpu' # for later use in torch.autocast
29 dtype = {'float32': torch.float32, 'bfloat16': torch.bfloat16, 'float16': torch.float16}[dtype]
30 ctx = nullcontext() if device_type == 'cpu' else torch.amp.autocast(device_type=device_type, dtype=dtype)
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

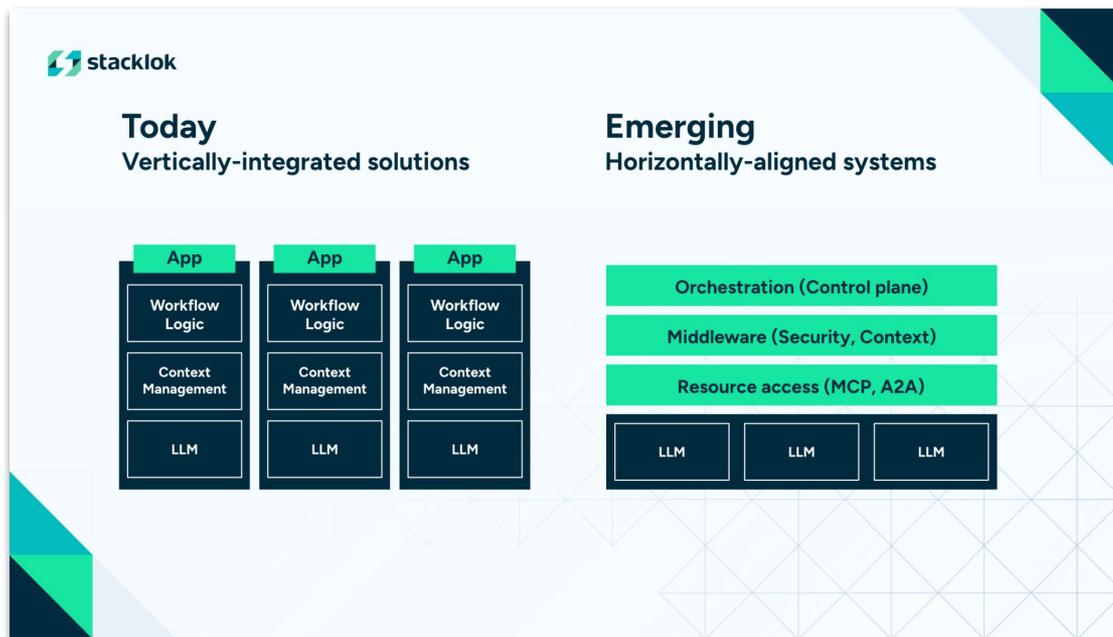
ty@Continues-MacBook-Pro-2 nanoGPT %



What is an AI-native open architecture?



An open architecture is a horizontally-aligned system



Slide from "Tipping AI Code Generation on its Side" at the O'Reilly AI Codecon, presented by Craig McLuckie, co-founder of Stacklok, Kubernetes, and CNCF

An open architecture is a horizontally-aligned system



The problems with ...
Vertically-integrated solutions

- Closed / opaque
- Subject to disruption
- Trust is outsourced

The power of ...
Horizontally-aligned systems

- Open / modular
- Community-led innovation
- Developer agency

The slide features a dark blue background with a light blue grid pattern. A white arrow points from the left column towards the right column. The Stacklok logo is in the top left corner.

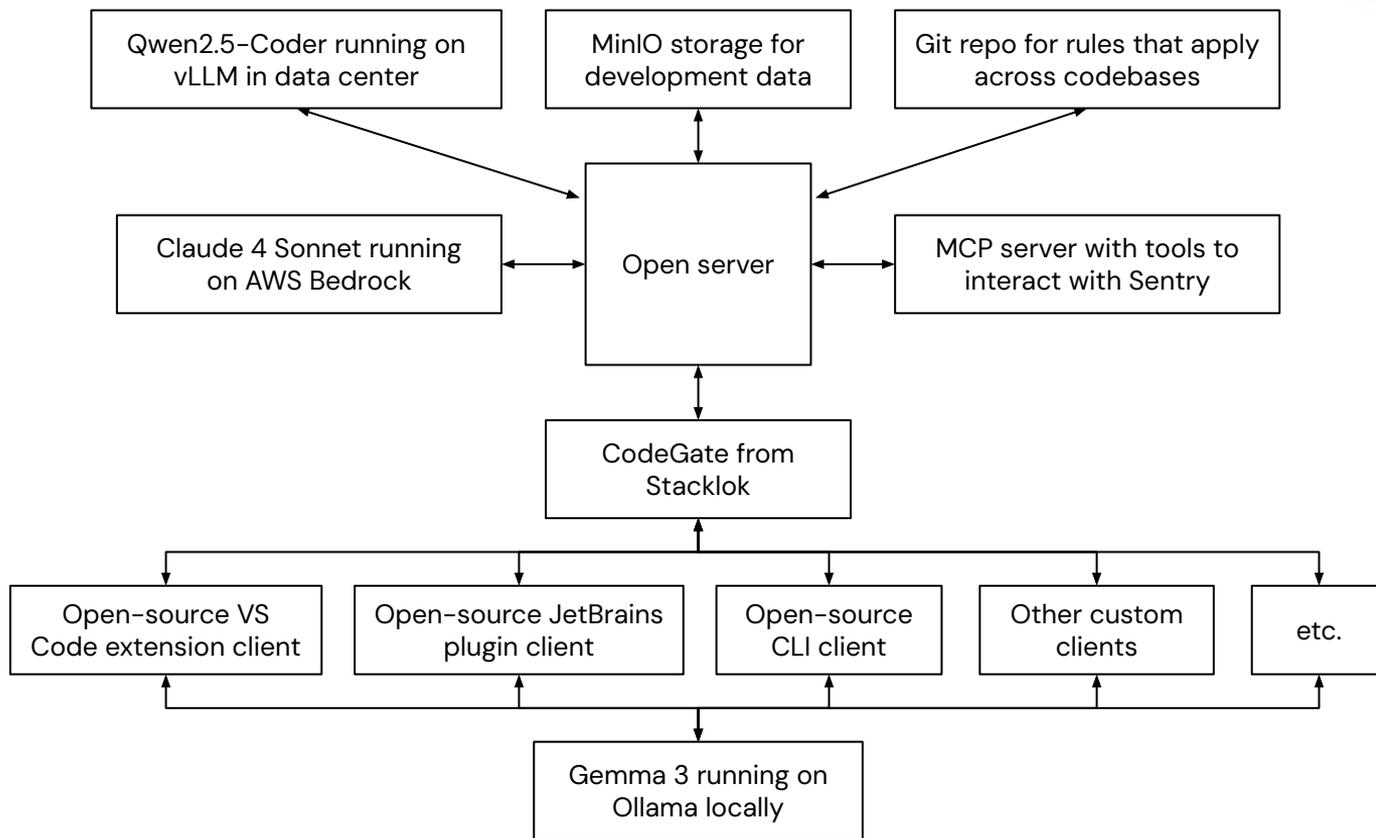
Slide from "Tipping AI Code Generation on its Side" at the O'Reilly AI Codecon, presented by Craig McLuckie, co-founder of Stacklok, Kubernetes, and CNCF



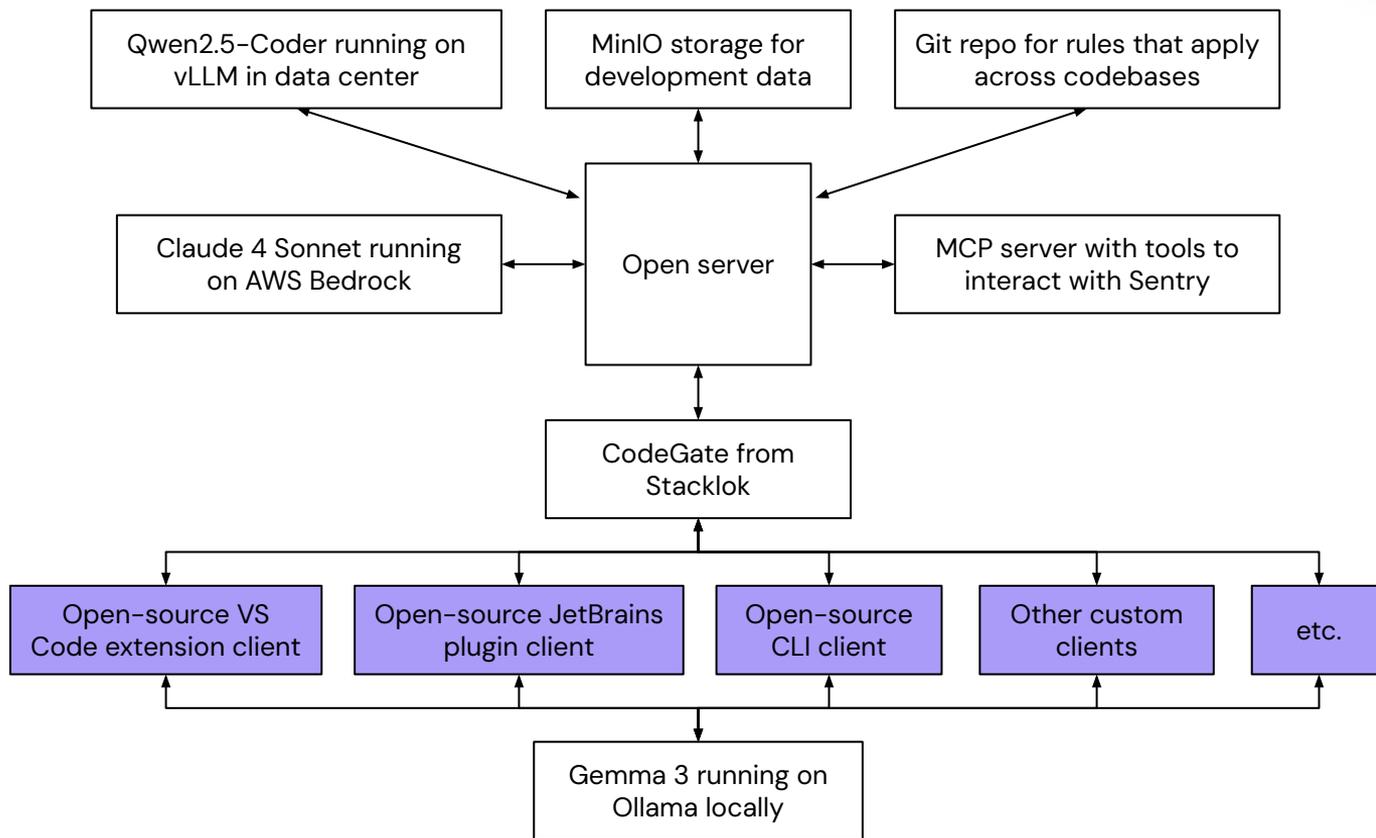
Overview of an AI-native open architecture



An AI-native open architecture



An AI-native open architecture: Clients



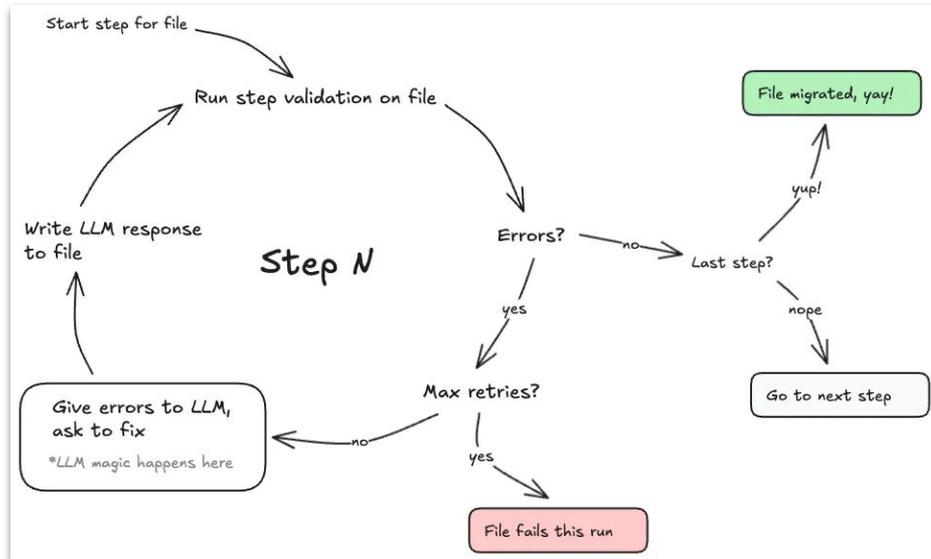
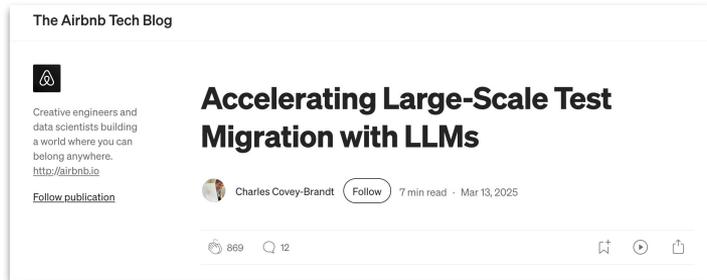
Open-source clients

- Examples
 - VS Code extension, JetBrains plugin, CLI tool
- Benefits
 - You understand how it works and can learn to use it better
 - Empower an ecosystem to build on top of and around them
 - Enables developer to keep and leverage their development data

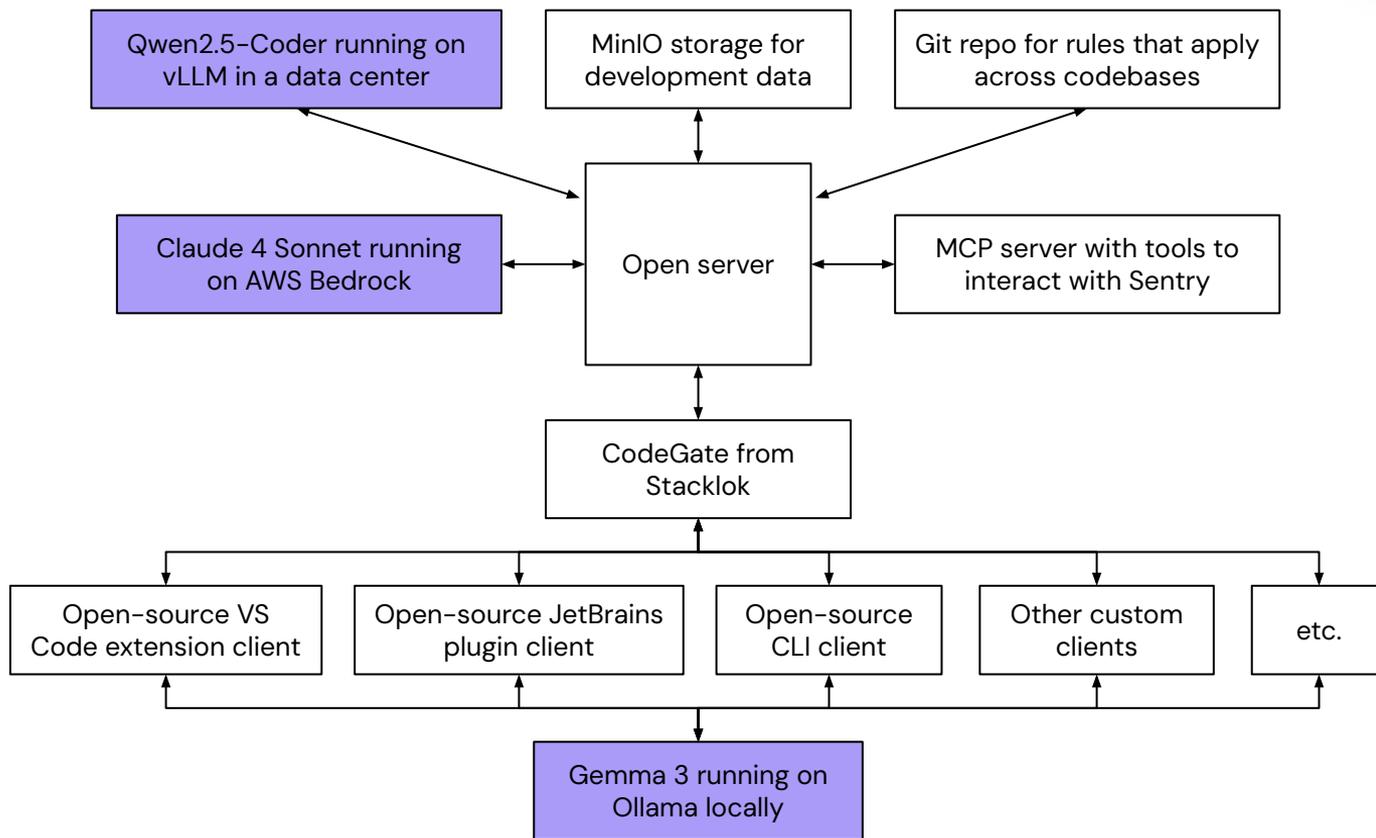
Custom clients

SDK / API that lets you pull in rules, models, tools into automation anywhere

For example, automation known as “agents” (rules, tools, and models in a loop)



An AI-native open architecture: Models



Models

Models are the intelligence that follows the rules using the tools to automate tasks. They provide the reasoning and tool use capabilities needed to implement automation.

Key Model Trends

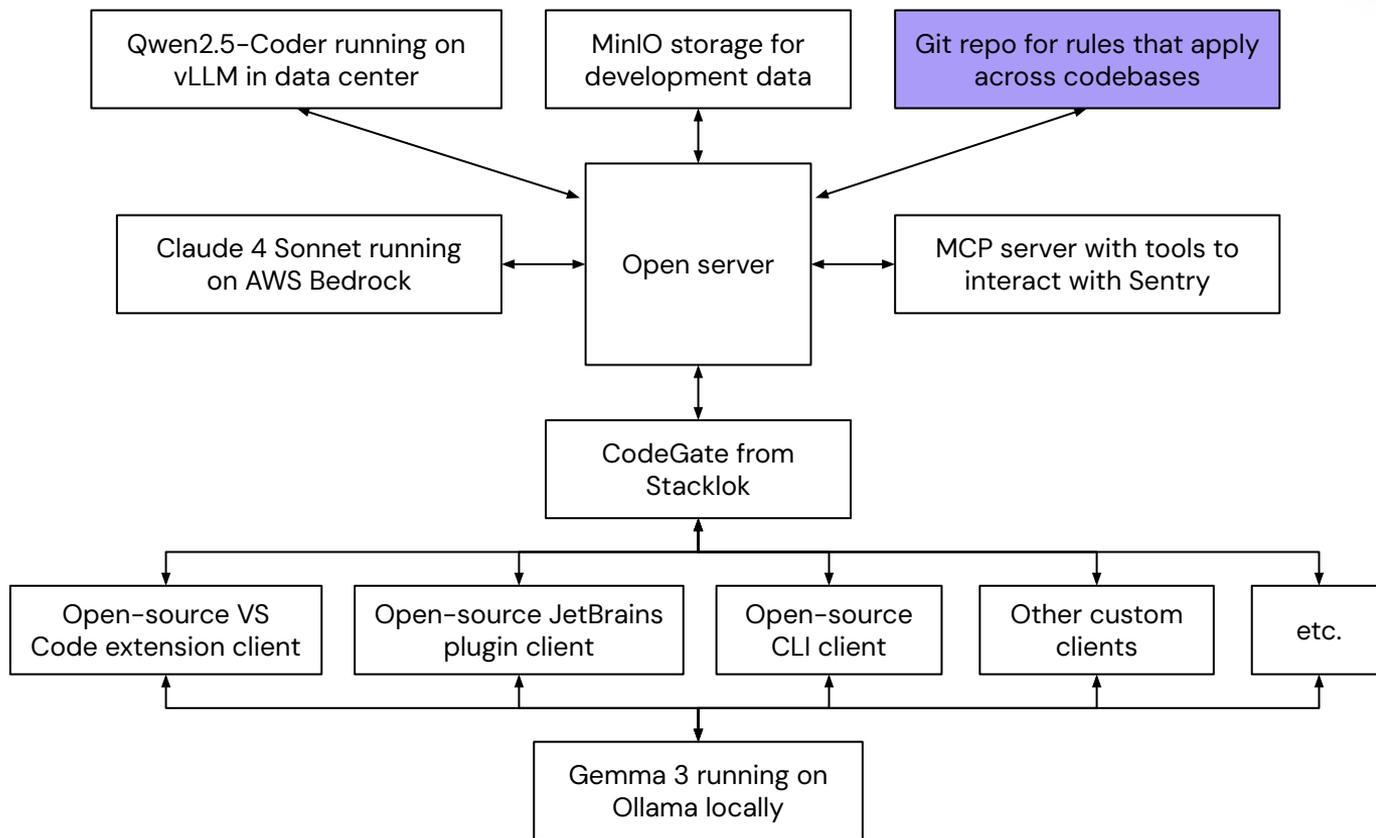
1. Increasing Capabilities
2. More capable open models
3. Improved economics
4. Diversifying task requirements
5. Growing domain specialization

Models distributed across compute environments

Specialized model deployments based on security, cost, and performance needs:

- **Local Development:** Efficient models running directly on developer machines for low-latency interactions
- **Enterprise Self-Hosting:** Support for organizations to fine-tune and deploy models behind their firewalls
- **VPC Deployments:** Secure cloud deployments within customer VPCs to maintain data sovereignty

An AI-native open architecture: Rules



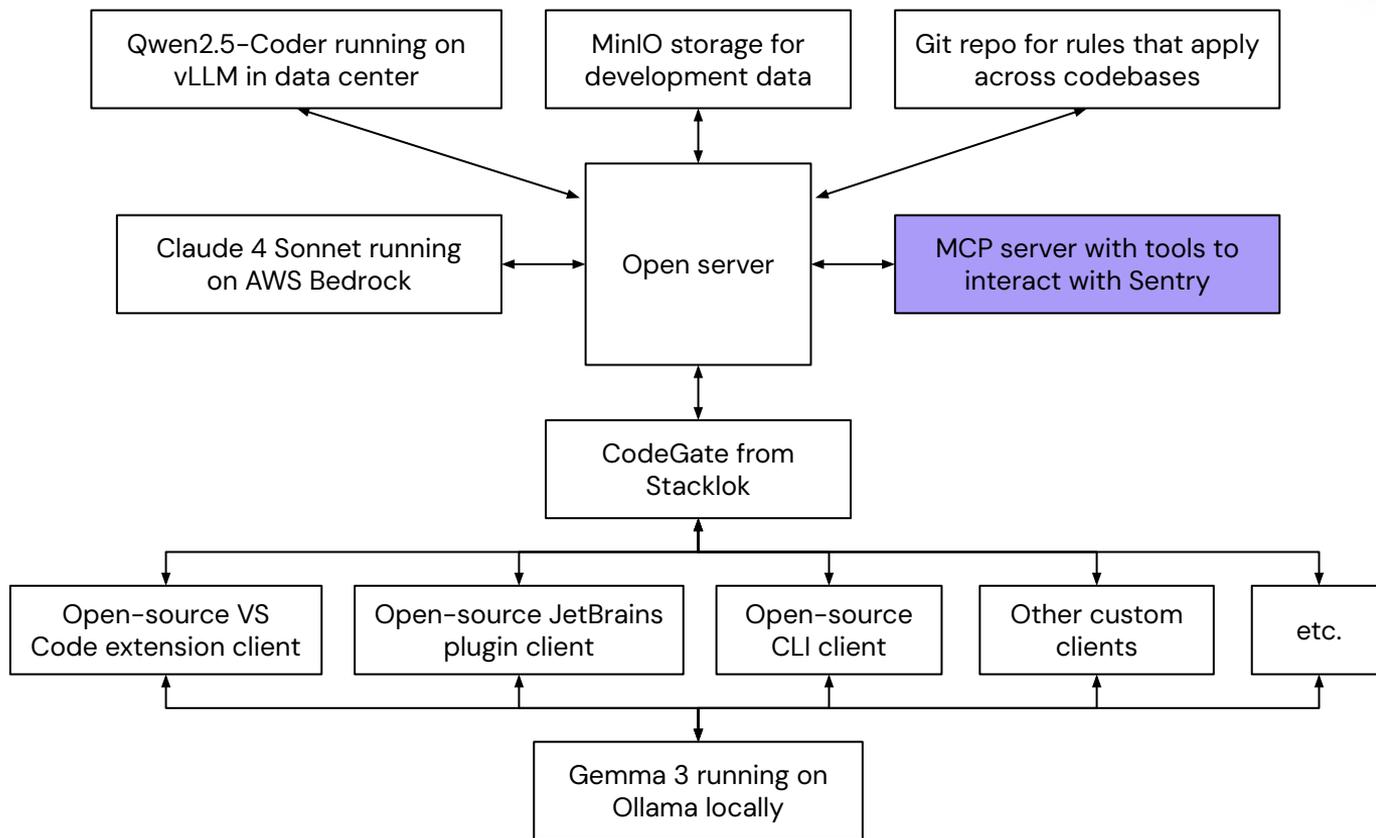
Rules

Rules are used to describe the **intents, constraints, policies, specifications, etc.** within an organization

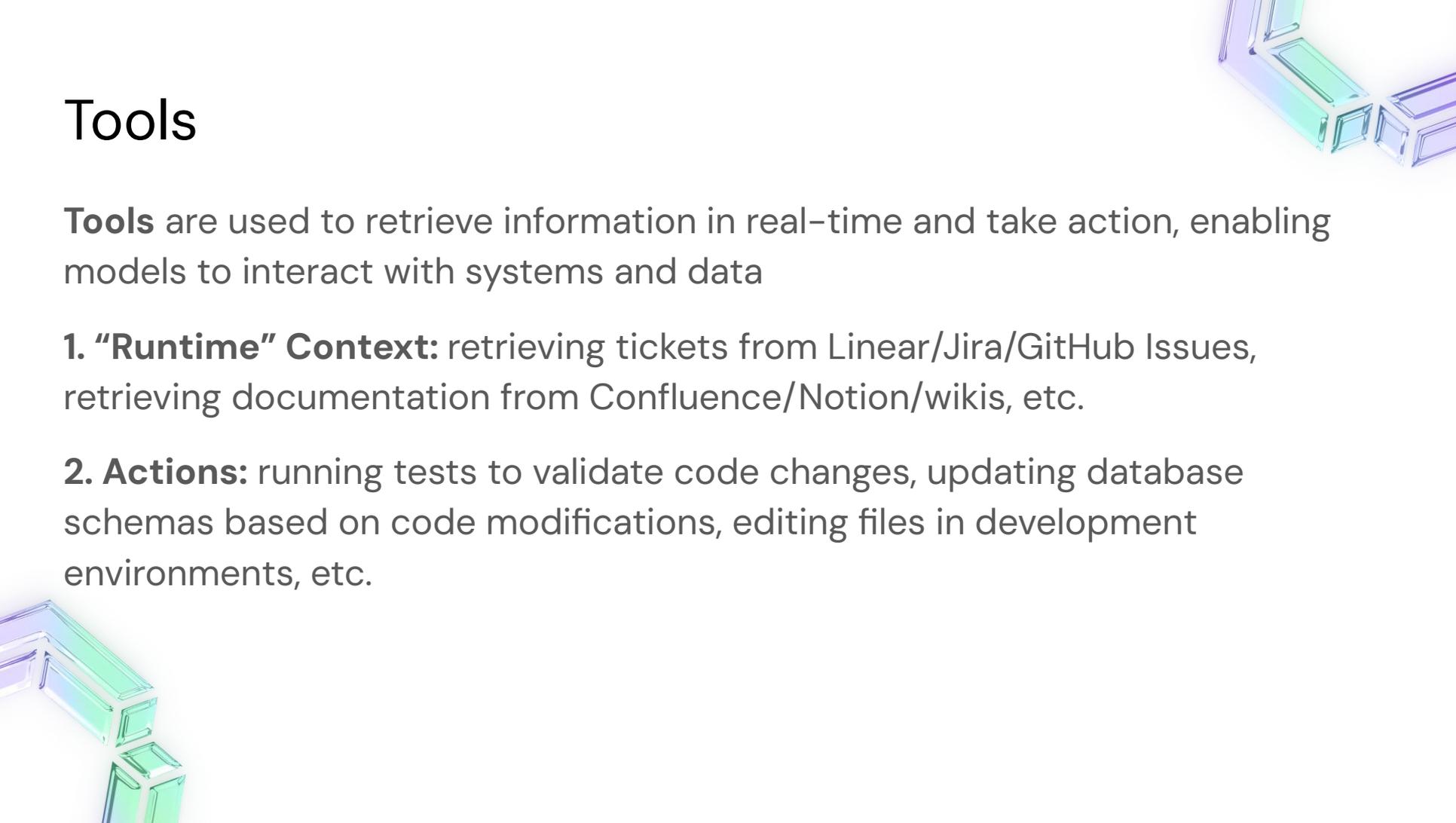
Collectively, they create comprehensive natural language descriptions that **make how you build software legible (i.e. rules) and accessible (i.e. tools) to models**

Some rules stored with codebases in Git, others that cut across codebases are stored in another centralized place (e.g. a Git repository)

An AI-native open architecture: Tools



Tools

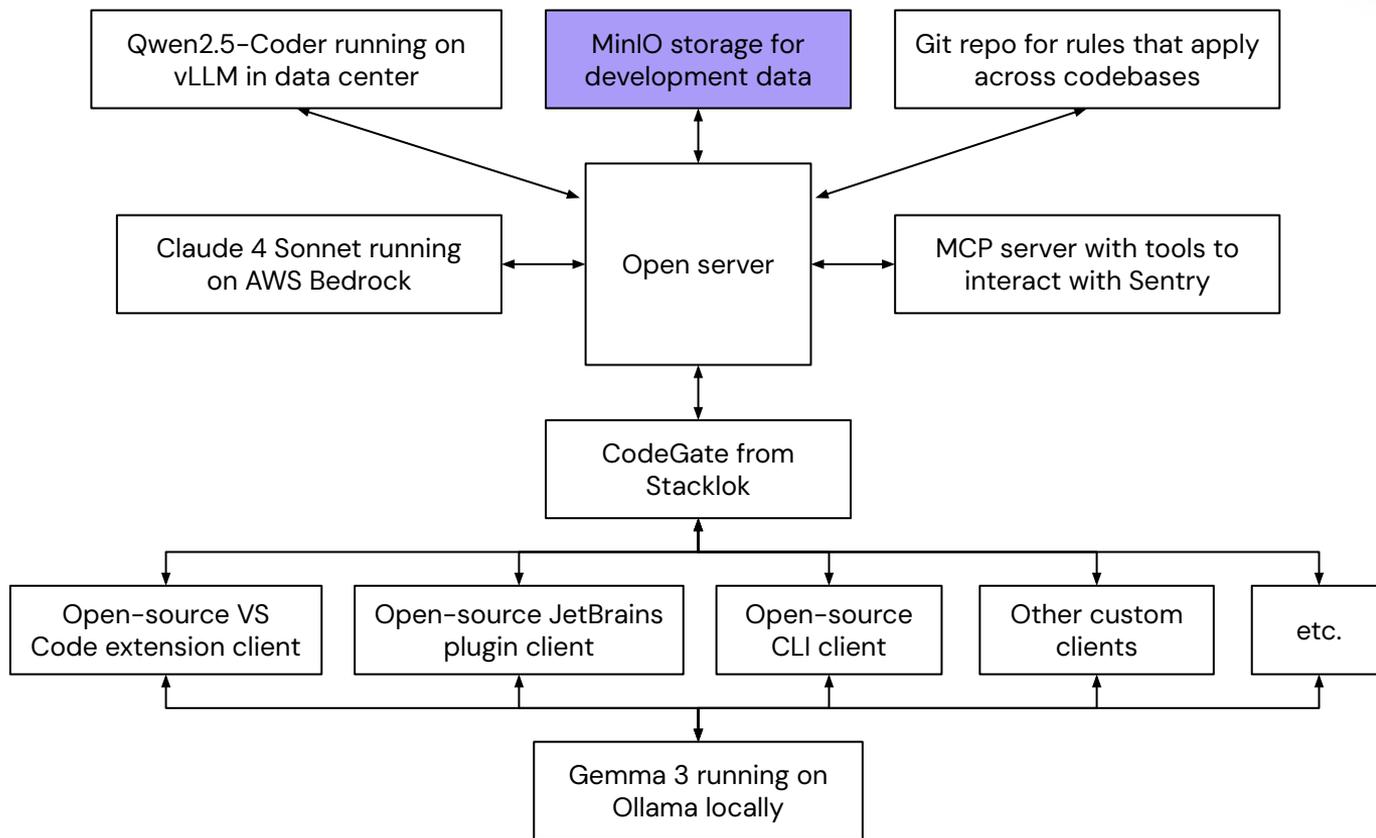


Tools are used to retrieve information in real-time and take action, enabling models to interact with systems and data

1. “Runtime” Context: retrieving tickets from Linear/Jira/GitHub Issues, retrieving documentation from Confluence/Notion/wikis, etc.

2. Actions: running tests to validate code changes, updating database schemas based on code modifications, editing files in development environments, etc.

An AI-native open architecture: Development data



Development data

Dev data captures **the "how" of development that exists between Git commits** and is automatically generated when you use models while coding

The **step-by-step process of software creation** that shows the

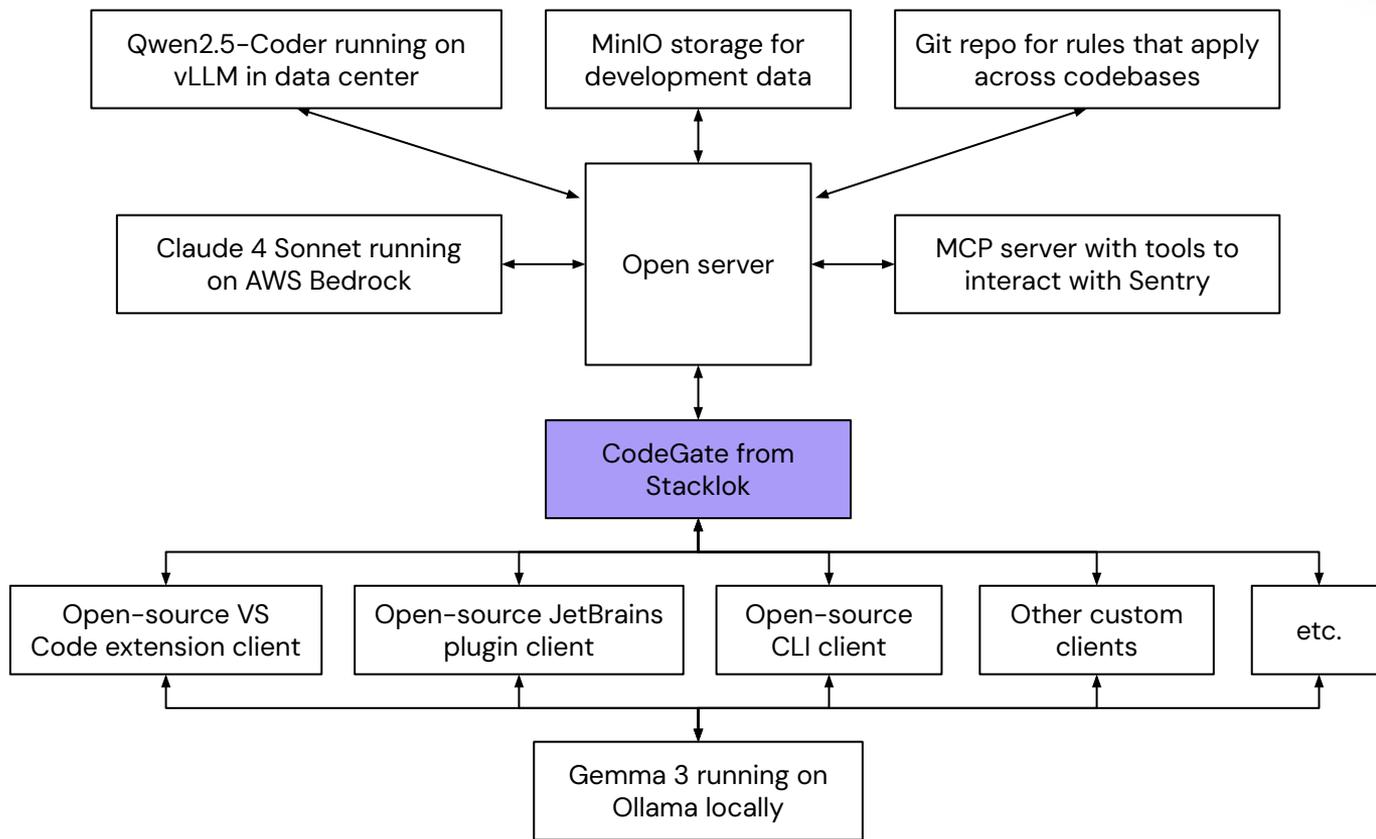
- patterns that make your organization unique
- context used to make development decisions
- reasoning behind each implementation choice

Development data

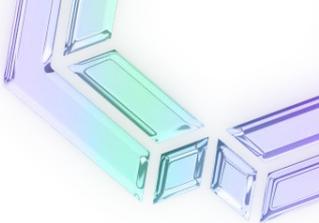
You can use dev data to **transform implicit, tribal knowledge into explicit, collective assets** that can be used to further automate development with AI

That is, development data can be used to improve rules, tools, and models

An AI-native open architecture: Ecosystem



Ecosystem

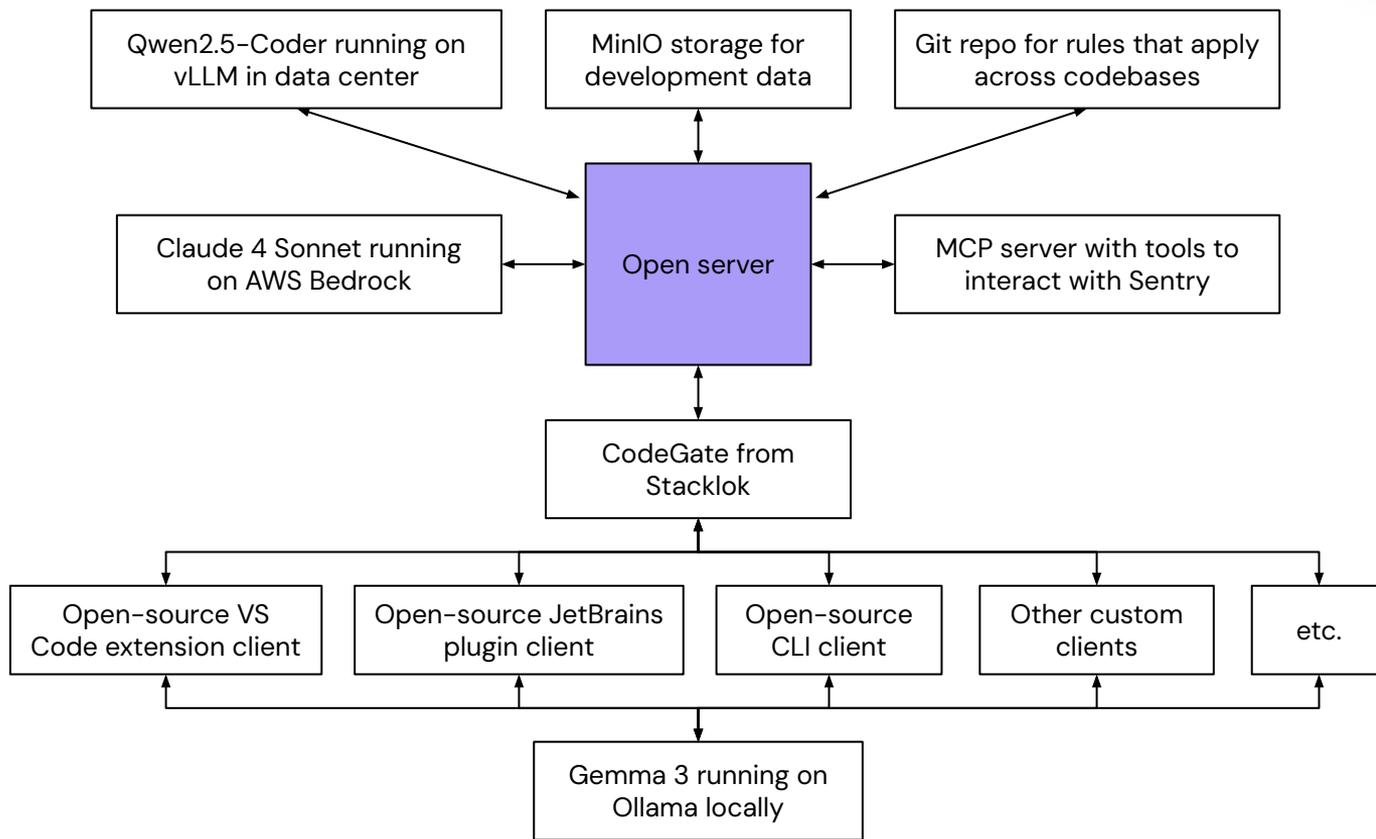


Anyone can extend open servers; an open architecture empowers an ecosystem to form and grow around it. Example:

CodeGate from Stacklok

- Secrets PII redaction to protect your sensitive credentials and anonymize personally identifiable information
 - Dependency risk awareness to update the LLM's knowledge of malicious or deprecated open source packages
 - One of many examples of to come; win together by working together
- 

An AI-native open architecture: Server



What is the AI-native open server now?

Examples:

- Authenticates users
- Proxies requests
- Manages configuration
- Enables observability
- Integrates with context sources
- Transforms development data before storage

What is the AI-native open server becoming?



- **Single source of truth** for how development works in an organization
- A natural language specification that **makes how you build software legible (i.e. rules) and accessible (i.e. tools) to models**
- This living description needs to be comprehensive, evolve with you, and require minimal human support / supervision

For this to happen, we need to design the required **1) workflows, 2) collaboration, and 3) governance**



1. Customization via the open server

Custom assistants to personalize the rules, tools, and models necessary for specific workflows. They make it easier for models to complete a task as desired by constraining and focusing the rules and tools. Examples:

- **Audit assistant** to audit for security vulnerabilities and create a report that helps you refactor and fix them
- **Plan assistant** to create a comprehensive product requirements document that can be used to guide implementation
- **Refactor assistant** to focus on improving existing code without adding new features

2. Collaboration via the open server

Collaborating with the organization and models to make **improvements of rules, tools, and models**

“Runtime” improvements (i.e. context windows)

- Enhancing rules utilization
- Improving tool integration
- Using dev data to improve context
- Better in-context learning

“Compile Time” Improvements (i.e. model training and fine-tuning)

- Enhancing via pre-training
- Optimizing via post-training
- Specializing for an organization
- Better domain-specific models

3. Governance via the open server

Enterprise requirements to provide the necessary guardrails. Examples:

- Security-first design
- Data sovereignty options
- Granular permission controls
- Policy enforcement
- Comprehensive audit trails
- Credential management
- Cost, performance monitoring

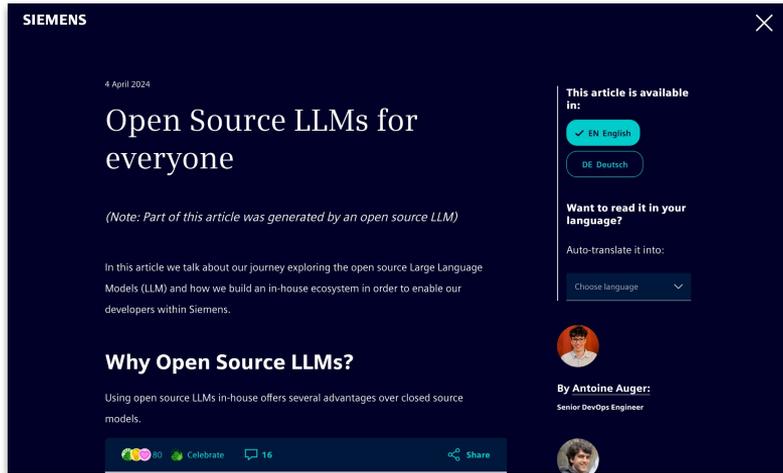


Why does an AI-native open architecture matter?



Why does an AI-native open architecture matter?

Creates significant advantages closed systems cannot match through **modularity, transparency, and interoperability**



SIEMENS

4 April 2024

Open Source LLMs for everyone

(Note: Part of this article was generated by an open source LLM)

In this article we talk about our journey exploring the open source Large Language Models (LLM) and how we build an in-house ecosystem in order to enable our developers within Siemens.

Why Open Source LLMs?

Using open source LLMs in-house offers several advantages over closed source models.

This article is available in:

- EN English
- DE Deutsch

Want to read it in your language?

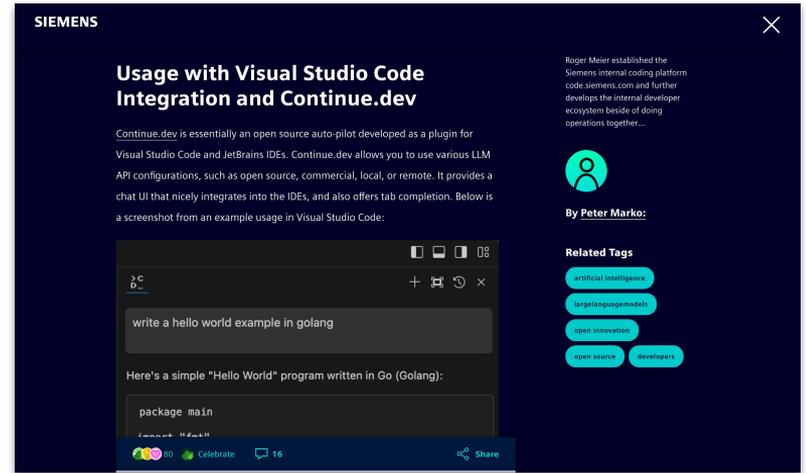
Auto-translate it into:

Choose language

By **Antoine Auger**
Senior DevOps Engineer

Celebrate 16

Share



SIEMENS

Usage with Visual Studio Code Integration and Continue.dev

Roger Meier established the Siemens internal coding platform `code-siemens.com` and further develops the internal developer ecosystem beside of doing operations together...

Continue.dev is essentially an open source auto-pilot developed as a plugin for Visual Studio Code and JetBrains IDEs. Continue.dev allows you to use various LLM API configurations, such as open source, commercial, local, or remote. It provides a chat UI that nicely integrates into the IDEs, and also offers tab completion. Below is a screenshot from an example usage in Visual Studio Code:

```
>C
D_
+ [ ] [ ] [ ] [ ]
write a hello world example in golang
Here's a simple "Hello World" program written in Go (Golang):
package main
import "fmt"
func main() {
    fmt.Println("Hello World")
}
```

By **Peter Marko**

Related Tags

- artificial intelligence
- large language models
- open innovation
- open source
- developers

Celebrate 16

Share

Why does an AI-native open architecture matter?

AI-native open architectures aim to amplify developers in the long run

- You keep your data to further automate your development with models

AI-native closed solutions aim to automate developers in the long run

- They collect your data to further automate you with models



Thanks!

Ty Dunn

Co-founder of Continue

ty@continue.dev

