

S-08, Anonymized sample from a real engagement

Final sprint report, sprint 18

A shortened closing report in 13 sections, with findings first, deploy faults fixed, validation counts and time.

Anonymized sample from a real sprint for a US investment analytics platform, 2026

What this is. The closing report of a nine day sprint, shortened for this sample. The real report has the same 13 sections, an evidence reference on every figure and a list of 77 commits, one plain sentence each. Names, fund tickers, vendors, commit hashes and amounts are replaced. Sections are cut where marked. Otherwise the wording is as sent, apart from small edits for house style.

Sprint 18 report

Started	Builds done	Tests	Time statement
[day 1]	8 of 8	683 to 922	54 hours

Sprint 18 built every item from your reply of [date] that can be built in code. The 3M return is on the fund cards and on each stock row. Long return tables keep their column headers in view. The drill down goes to 1,800 rows. Staff can rerun scripts for an earlier date. The ad network code is in place. The code went live on [day 2], this sprint and the earlier scope together.

Reruns and ads stay switched off until the sandbox and your ad account exist. I applied most of the infrastructure plan the same day. The secrets step, the sign in role and the sandbox still wait for you. One finding from production comes first, because it affects your reports every night.

Your scope listed sixteen items and your analyst added one. Eight of the seventeen are builds, and I built and tested all eight. Eight items are steps on production. The code deploy and the disk change are done. Most of the Terraform apply is done too. The other six wait for your go ahead, and for three of them the groundwork is done here. The last item is this report.

The merged code was then reviewed in two separate passes. A validation pass checked the figures in the sprint documents against raw data, 388 statements in all. And a fresh local copy, loaded with real holdings and real prices, ran every new page in a browser. Figures carry a reference in square brackets to the evidence listed in section 13.

1. The nightly pipeline stops part way, read this first

Found on [day 1], needs your decision.

I was looking for the reason the nightly run grew from 40 minutes to 2 hours. The step timings showed something worse. The pipeline stopped part way on 14 of the 30 nights still in the log. On those nights the run ends inside the script step, after 190 to 230 scripts. The steps after it never start. Those are the retry of failed scripts, the report rebuild and the report emails. [E9]

Memory is the likely cause, and it isn't proven yet. Each of those nights the log ends within six minutes of the batch that holds the two heaviest scripts. One run of each in the web container raised memory there by about 1.2 GB and 1.5 GB. The pipeline task has 4 GB and starts ten scripts at a time. So I'll read the task's exit code right after a run, while the cloud provider still holds it. [E9]

Update of [day 9]. A watcher read the task record within five minutes of the end of the run of [day 2]. It showed that the container was stopped for lack of memory, with exit code 137. The run stopped in the same step on every night since. [E22]

The script sandbox is the real fix. Each script then runs in its own task with its own 1 GB, and one heavy script can't end the run for the others. Three small changes would help before it, and none is in this sprint's hours. A default timeout on a script's outside requests is about 2 hours. One shared deadline for a batch of ten is 1 hour. Recycling the web workers after a set number of requests is half an hour. Tell me if you want them.

The script scan for section 7 turned up two more things. 197 of the 342 scripts were in failure status, and 88 of those can't run as stored. And 72 scripts carry what looks like a third party API key written into their code. Until [day 2] any signed in user could read script code on the live site. I built the change that limits script code to staff and the owner and deployed it on [day 2]. The keys should be replaced, with the new ones kept in the secrets store. [E10]

One page on the live site showed sample data as if it were current. It read its rows from a script that draws a chart and stores no table, so it fell back every time to thirty built in sample rows from two years earlier and stamped them with the current time. I built the fix and deployed it on [day 2]. The page now says the returns aren't available and shows no figures. [E17]

2. Your analyst's fund prices

Live since [day 1]. He asked for the prices of 27 world fund tickers through the data helper. Production held no row for 19 of them. The platform now prices the fund behind every holdings list it reads, with ten tests. Each of the 27 returns 2,960 daily rows from January 2015. The load inserted 56,240 rows and changed none. [E8]

3. The 3M return on the fund cards

Live since [day 2], your point 1. Three months is the close 63 trading sessions back, the same rule the cards now use for one week and one month. It's a total return, like every other card figure. I checked it against an independent recompute from public closes for three funds. The platform's own calculation gives the same total return and price return figures to four decimals, six in all. [E11] [E18]

The card email grew by one column. I narrowed the cell padding, so the table is 794 pixels wide where it was 801.

4. The 3M return on each stock row

Live since [day 2], your point 2. Every stock row in the drill down carries the same 63 session figure, in the table and in its CSV. It's a price return, like the other figures on those rows, and the table says so. Three named stocks and 60 of 60 rows picked at random each equal a separate recompute. [E11] [E18]

5. Column headers that stay in view

Live since [day 2], checked in a browser. The header row now stays at the top while the rows move under it. On a phone the ticker column also stays at the left while the table scrolls sideways. One thing looks different. A long table now scrolls inside its own box, at most three quarters of the screen high, where it used to lengthen the page. The old code had a sticky rule that never worked, because the table sat in a sideways scrolling wrapper.

A browser spec drives the real pages with stubbed data at a desktop width and at a 375 pixel phone width. It runs 7 checks and all pass. One is a control that removes the new rule and watches the headers scroll away. I haven't held it on a physical phone, so I still owe that check. [E12] [E18]

6. The drill down raised from 600 to 1,800 rows

Live since [day 2], your point 7. I corrected three numbers I gave you earlier by measurement. The nightly cap goes to 11,000 names and not 10,000. The new names are 3,355 and not the 4,137 I estimated. And the page reads about 35 times fewer price rows, not 45. [E13]

The table now pages, 100 rows at a time. Sorting and search run on the server, so they work across all 1,800 rows and not only the page on screen. The CSV still exports every row. The server compresses the answer for the first time. One page of 100 is about 6 KB compressed, where the live page was 195,598 bytes for 600 rows with no compression. [E13] [E18]

The run on real data found two faults in the new pager, and I fixed both. Fast paging met the site wide limit of 30 requests a minute in 13 seconds, so the drill down now has its own allowance of 120. And a click on Next in the first third of a second was undone by the search box's delay timer. With the fix the click holds in 20 of 20 tries. [E18]

7. Change the date and rerun a script

Built, waits for the sandbox. Staff get a panel on the script page with a date picker, a Rerun button and the run history. A rerun keeps its own chart and table, so the daily chart is untouched. Every one of the 13 data helpers stops at the date you picked, even when the script asks for a later one. And the script's clock reads that date, across all five clock forms your scripts use.

What a rerun can't do. Data from an outside site is today's data, and the rerun says so with the sites named. No rerun has run in a real task yet. The first real rerun runs with the sandbox shadow run.

What a script needs before a rerun is right	Scripts	In success status
Nothing. Platform data only	34	16
Nothing, by my reading. Outside data whose end date already comes from the clock	104	67
The as of date passed to its outside request, 0.5 to 1 hour each	114	62
Can't be rerun as stored	89	0
Couldn't be read	1	0

That's a prediction from the code. One real rerun per outside site will confirm it. Tell me which scripts you want to rerun and I convert those. [E10]

8. The ad network, my side

Built, off until your publisher ID. With no publisher ID the site shows no ads and loads no ad code. The built page is byte for byte the same, which a test checks. The privacy page is real now, open without a sign in. Ten points in it are statements that are yours to approve, and the legal name, jurisdiction and effective date are blank fields for you to fill. I didn't invent anything a lawyer should decide. [E14]

9. The cloud bill, the nightly run and the web memory

Measured, the bill match waits for billing access. A public statistics site began to stall connections, and the scripts that read it have no timeout. Each stalled script holds its batch of ten for 10 to 16 minutes. 155 of the 180 timeouts come from 22 macro scripts. The web memory jump lines up with the first day anyone pressed Run. Every later step of more than 5 points lines up with a Run too. The sandbox moves those runs out of the web container. [E9]

10. The database disk and the pipeline's own sign in

Disk applied on [day 2], the sign in role waits. The disk went from 10 GB to 20 GB on faster storage, with autoscaling up to 50 GB, in one in place change with the database online. Its baseline went from 100 to 3,000 operations a second. I wrote the deploy pipeline's own sign in role too. It can do what the ten routine workflows need and nothing more. It waits on one repository setting that only an owner can make. [E15]

11. Two guides for you

Written, your point 14. I wasn't sure which you meant, so I wrote both. One shows how you put an API key into the secrets store yourself, so the value never passes through chat, email or me. The other shows how to add a passkey to your cloud sign in. Three button names in the first guide I couldn't confirm against a document, and the guide marks them. [E15]

12. The deploy, and what's waiting on you

I deployed on [day 2]. I took a database snapshot first. The backend went out before the frontend. Six database migrations ran and ended without an error. No Terraform apply ran with the code deploy. [E19]

The deploy found four faults, and I fixed all four the same day. Four frontend pages joined the API address and a path without a slash. Three browser checks read the page too early. One query asked for the newest close date with no floor, so the database read the whole price table, 15 million rows, in about 7 seconds. It now looks at the last fourteen days first and takes 0.06 seconds. After that fix a fund's first load went from 8 to 12 seconds down to 1.7 to 4.7 seconds. And the page went out with no caching rule, so a browser kept its old copy for a day. [E19] [E20]

I read the live pages in a browser at a desktop and a phone width, signed in and signed out. I then checked 41 statements against the signed in site, field by field. All 41 held. [E20]

What I need from you, in order of urgency. Your go ahead on the secrets step and a quiet half hour. A yes or no on the three small pipeline fixes in section 1. The main only rule on the two production environments in GitHub, which only you can set. Permission for the Terraform user to create a secret, which the sandbox needs. Billing access or a cost export. Your approval of the privacy page text. And the ad network IDs when the account exists.

Review and validation

The backend review found that a script switched off could still be rerun, and fixed it with three smaller faults in the rerun path. It ran the migrations on PostgreSQL 15, forward, back and forward. The validation pass checked 388 statements, and 291 held as written. A third round after the deploy checked 441 statements against the live site, the cloud account, GitHub and the code. 363 held as written, I corrected 70 here and I can't check 8 from here. [E16]

I left three small things open and wrote them down. Two staff asking for a rerun in the same instant can pass the platform limit of four by one. One script imports a package the rerun clock can't carry yet. And two older tests can't run on PostgreSQL because of their test data. [E16]

Time

This sprint's statement is 54 hours, each item at the hours I quoted or inside the quoted range. I don't charge two things. The scan of all 342 scripts, which I'd quoted at 2 hours. And the work of joining the builds into one branch, the two reviews, the validation and the run on real data. The four fixes of the deploy day belong to that too. Those are integration costs, not work you asked for.

13. Sources for every figure in this report

[Shortened for this sample. The real section is a table of 15 evidence references. Each names the files in the evidence folder and what they support. One example row follows.]

Ref	Source	What it supports
E13	Universe count on production, the closes and payload measurement, the live response headers	5,717 and 9,072 names counted on production, the rows read before and after, the payload sizes, the missing compression on the live page

[The commit list follows, one plain sentence per commit. Three examples.]

- Ask for a fund's newest close date with a floor so PostgreSQL stops reading the whole price table
- Give the fund drill down its own allowance of 120 requests a minute so paging no longer meets the site wide 30
- Keep the reader on the page they chose when Next is clicked before the search delay has passed

Prepared on [day 1] and [day 2], with updates on [day 9]. I read every measured figure here from a live run, from production's logs and settings, or from the test suite. I read the logs and settings without changing anything. I marked estimates as such.

Axdale, Michael Lip. Warsaw, Poland (CET). michael@axdale.com.
www.axdale.com. Code record github.com/theluckystrike. Names,
tickers and amounts are replaced.