

S-03, Anonymized sample from a real engagement

Sprint 18 scope

The written scope for a nine day sprint, with measurements, hours per item, parked items and what the client had to do.

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

What this is. The written scope I sent on day 1 of sprint 18. The client agreed it in writing before the build started. Names, fund tickers, vendors and amounts are replaced. The structure, the hour estimates and the level of detail are as sent.

Sprint 18 scope

Prepared [day 1]

Sprint 18 carries three things. It answers your cloud bill question. It finishes what you approved on [date] and sprint 17 didn't reach. And it takes the three items you assigned to sprint 18 that day, the 1,800 rows, the date rerun and the ad network.

Where sprint 17 stopped. It delivered your analyst's three requests, and those are live. The scope of [date] is built and tested on a branch, with 683 tests passing. Nothing in it is live yet, because several steps change production and need your go ahead. The 3M returns and the locked column headers aren't built yet. All of it is in part 1 below.

Your cloud bill question, what I measured today

My cloud login can read usage but not the bill itself, so I measured every metered resource from June to [date]. The recent changes add a few dollars a month at list prices.

- Nightly jobs. Since [date] the nightly work runs as one pipeline task with 2 vCPU and 4 GB. Scheduled jobs ran 12 to 13 task hours a month from June to August and 48 hours in September. The sprint 17 price work didn't lengthen the nightly run.
- Database. The instance is the same size, the processor averages 4 percent, and no burst credits were charged in any month. The allocated disk is still 10 GB, so the storage charge didn't move. Data on it grew from 2.2 GB to 4.6 GB.

- Backups. Since [date] the database keeps 14 days of backups, up from 7, and I took 5 manual snapshots before production writes. I can delete the 4 older snapshots.
- Everything else. Logs, the load balancer, file storage and the network gateway each moved by cents. I found no servers, databases or gateways in any of 8 other regions.

If the bill rose by more than that, the cause is outside these changes, and the billing view will name it in minutes. I list the two ways to give me that view at the end.

Two things the same check turned up

- The web container's memory peak sat at 12 percent until [date]. It has run between 61 and 76 percent since, and it reached 99 percent on [date]. A full container restarts and the site drops for a minute or two. Uploaded scripts run inside that container today, which is the likely cause, and the sandbox moves them out.
- The database disk has a small speed allowance at 10 GB. It fell to 25 percent on [date], where it normally sits at 99. At zero, every page that reads prices slows down.

Part 1, the cloud bill and everything carried over from sprint 17

Item 1, cloud bill, closed out, about 2 hours

Once I can read the bill, I match the measurements above against it by service and by day. I also find out why the nightly run grew from 40 minutes to about 2 hours on [date]. You get a one page answer with the numbers.

Item 2, the exposed secrets closed, about 3 hours

One old test script prints four secrets into its error text every night. They are the database password, the cloud key, the Django key and the cache address. I checked today, and the log shows it once a night for the last ten nights. I delete the script's stored error and switch the script off. Then I rotate the database password, the Django key and the cache password, and delete the affected log streams. The Django key rotation signs every user out once. I'd do this one first, with your go ahead, in a quiet half hour.

Item 3, the earlier scope deployed, about 4 hours

The deploy takes three database migrations, a Terraform apply that I read line by line before it runs, and the deploy workflow. It puts live the corrected one week and one month figures, the total return labels, a second exchange list and the morning health email. It also puts live the redacted script errors, sub industry data for your analyst and the tidied site. I take a database snapshot first, and I watch the first night's price fetch and the first health email. One fund's holdings from a second issuer are built too. You said on [date] that it can wait, so it stays off unless you tell me otherwise.

Item 4, the static cloud key retired, about 3 hours

The deploy removes the key from all thirteen task definitions. The key itself stays active until the deploy pipeline has credentials of its own, since the pipeline almost certainly uses the same key. I give the pipeline GitHub's short lived sign in, switch the key off for a day, confirm nothing breaks, then delete it.

Item 5, the script sandbox, shadow run and switch on, about 5 hours

I create the read only database login, run one script in the sandbox, then run all 345 beside today's results and read every difference. Six scripts use the platform's own API and get a token of their own. Then I switch the sandbox on and watch the first night. Switching it off again is the rollback.

Item 6, reports, your analyst's rights and the health email address, about 1.5 hours

11 of the 13 reports fail every day because the May recovery lost their script links. A command reads each report's PDF and links them back, and it prints its plan before it changes anything. Your analyst's script rights move to his email login with a second command once you agree.

Item 7, 3M returns on the fund cards, about 7 hours

The close 63 trading days back, measured the same way as the week and month after their fix. It covers the card, the detail table and CSV, the email and PDF, and the tests.

Item 8, 3M on the individual stock rows, 2 to 3 hours

The same 3M figure on every stock row in the drill down. The 2Y, 5Y and 10Y annualised figures stay out, as you asked.

Item 9, column headers locked on long return cards, about 2 hours

The header row stays in view while a long table scrolls, so each column keeps its label. I check it on a phone and on a desktop browser.

Item 10, database disk to 20 GB on faster storage, about 1 hour

The disk goes from 10 GB to 20 GB on the gp3 type, with storage autoscaling switched on. That fixes the speed allowance above and leaves room for the 1,800 row history load. The change runs with the database online.

Item 11, a passkey for your cloud account, about 0.5 hour

The step by step guide I promised on [date], with the exact button and menu names for each step.

Item 12, web container memory, cause confirmed, about 2 hours

I trace the jump seen on [date] to its source and measure the container again after the sandbox is on. If scripts were the cause, the peak falls back and nothing else changes. If not, you get the cause and the fix with its hours.

Item 13, report, compliance sheet and time statement, about 3 hours

Part 1 total, 36.5 hours.

Part 2, the items you assigned to sprint 18 on [date]

Item 14, drill down raised from 600 to 1,800 rows, about 9.5 hours

The cap goes to 10,000 names, the three 600 limits go to 1,800, and the six large funds price up to 1,800 names each. The page reads only the closes its returns need, about 45 times fewer rows per load, and the table pages. A one off history load of 1 to 2 hours fills the new names. The disk step is item 10.

Item 15, change dates and rerun a script, 16 to 17 hours

An as of date passes into each run, and the data helpers never return data after it. Each rerun keeps its own record, so the daily chart stays untouched. Staff get a date picker with a Rerun button and the run history. This builds on the sandbox, which is why it comes after item 5. Conversion per script is separate, and a 2 hour scan of all 345 scripts gives you exact counts first.

Item 16, ad network integration, 8 to 10 hours

My part covers ads.txt, the ad tag, a real privacy page, 6 ad units that collapse when empty, consent defaults and QA. It starts when you send the publisher ID, and real ads appear after the network approves the site.

Part 2 total, 33.5 to 36.5 hours.

Both parts come to 70 to 73 hours. I'd deliver part 1 first and part 2 the week after, so the security work and the 3M returns don't wait for the larger builds.

Ready to add if you want it

- API keys moved to a managed secrets store, 6 to 8 hours. It should land before the new data APIs on your list.
- Script code visible to staff and the owner only, about 1 hour. I raised this decision on [date].

Parked, as you decided on [date]

- Market cap as a field
- 2Y, 5Y and 10Y on the stock rows
- An as of date, fallback and changes endpoint for one sector fund
- An issuer file cross check against the public index list
- Point in time index history and purchased data

What I need from you

First, your go ahead on item 2, and a half hour when a single sign out of all users is acceptable. Second, a way for me to read the bill. That's read only billing access for my user, or an export of the cost report by service and by day from [date].

Then your agreement that the deploy pipeline gets its own credentials, so the static key can go. Your analyst's email address, and your agreement to move his script rights. The address for the morning health email. And whether the second issuer source goes live with the rest or stays off.

Later, your ad network publisher ID when the account exists. The scripts you want to rerun on past dates first, so I convert those and not all 345. And a yes or no on the secrets store and on the script code change.

Michael

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