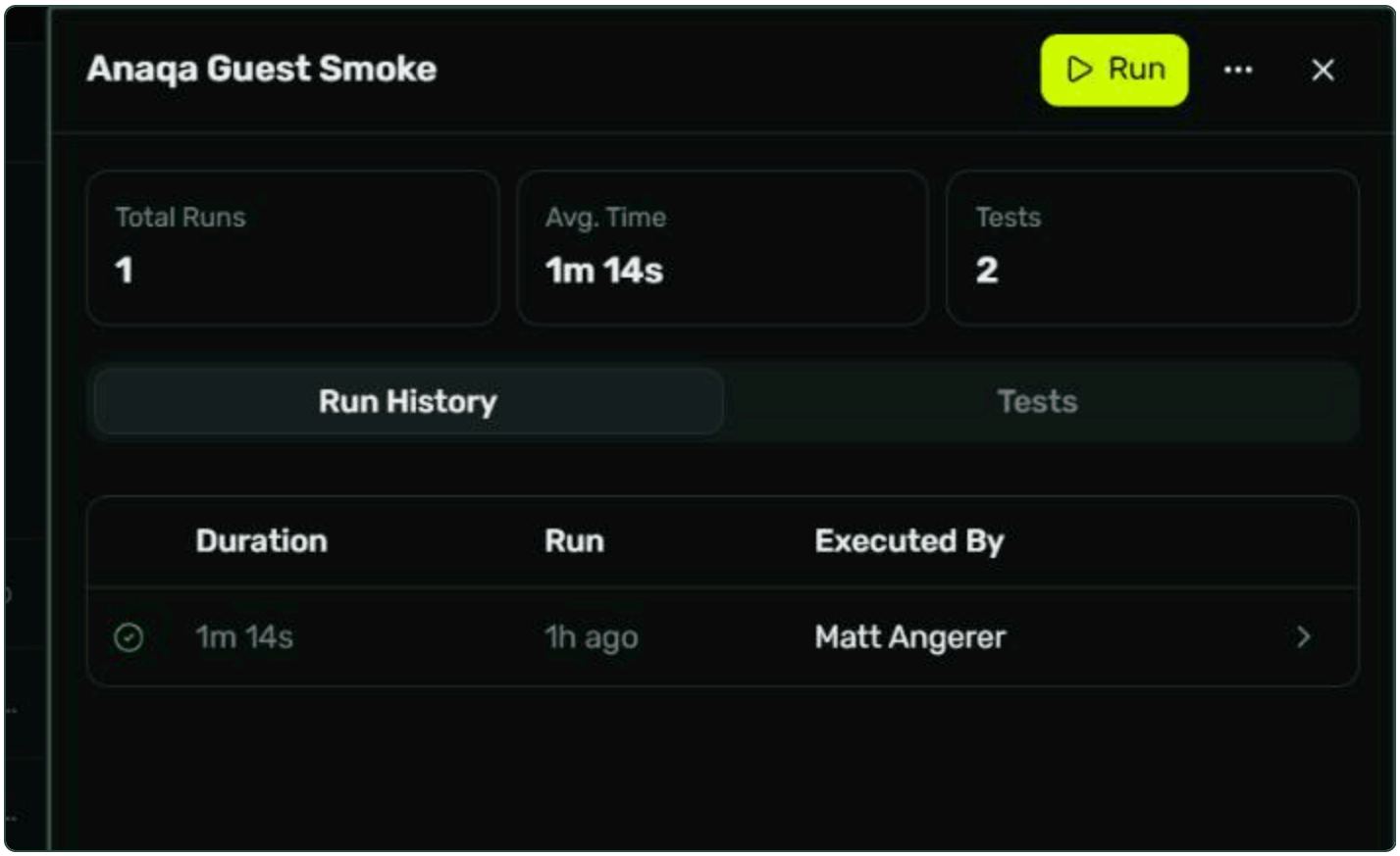
Orchestrations: Running a Suite as One Job

Features · https://docs.functionize.com/hc/en-us/articles/43789244334359-Orchestrations-Running-a-Suite-as-One-Job

A test tells you about one flow. An orchestration tells you whether the application still works. It is an ordered group of tests that runs as a single job, on demand or on a schedule, with one result and one history.

This is the feature that turns a collection of tests into something a team relies on. Until a suite runs on a schedule and somebody reads the result, it is a set of files.



What an orchestration is

A group of tests	Drawn from one project or from several — the field is **Projects**, plural, and it holds more than one. They run as one job rather than as several unrelated runs.
A schedule	On demand, or on a cadence. On demand is the right starting point; move to a schedule once the suite is stable enough that a red result means something.

One result	The job passes or fails as a unit, and the run history records the whole job — its result, duration and who triggered it. Individual test results stay reachable underneath.
A name that ages well	Name it after what it proves, not when it runs. *Guest smoke* survives a schedule change; *Nightly 2am* becomes a lie the first time somebody moves it.

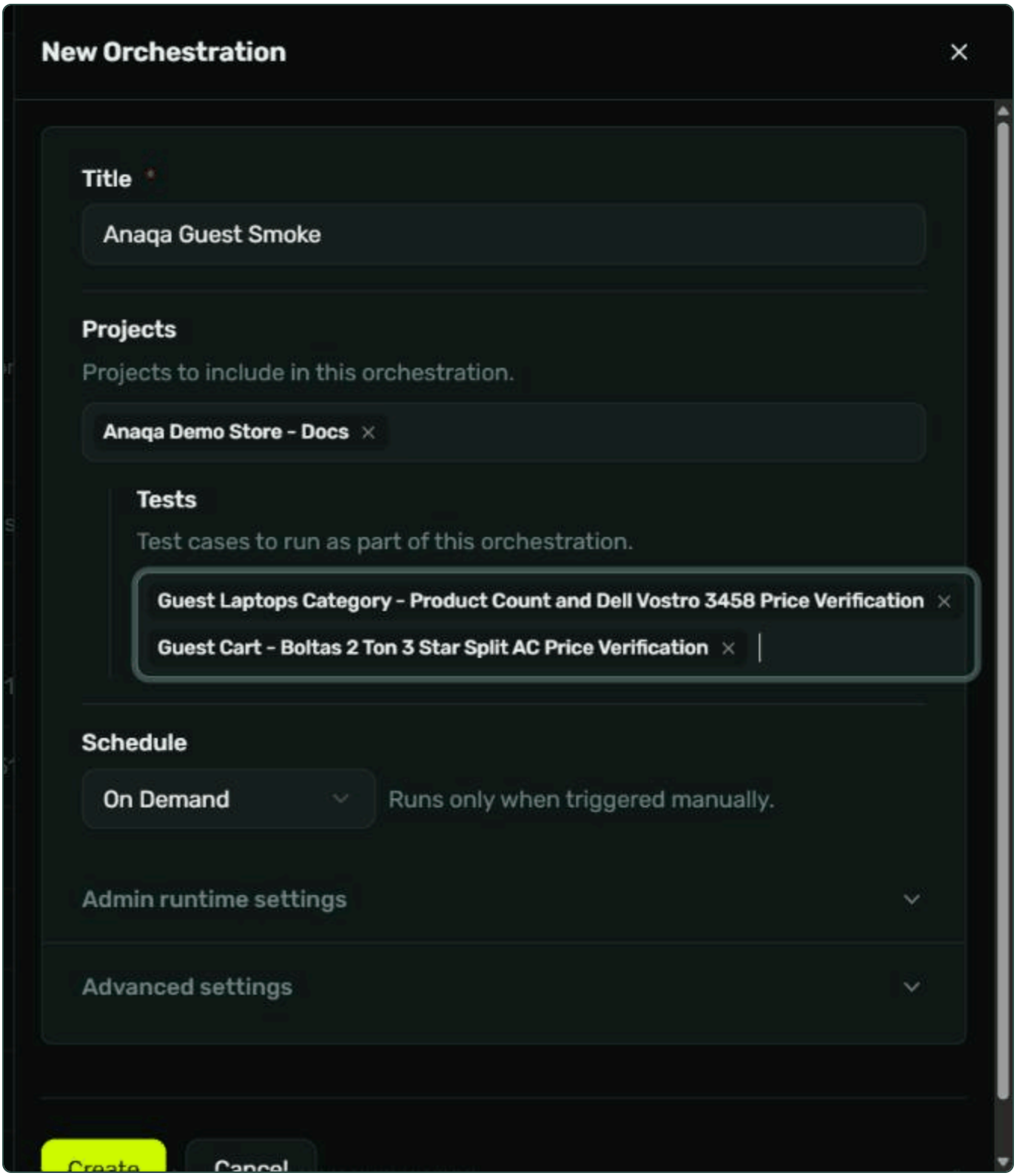
Creating one

Open **Orchestrations** in the left sidebar and choose **New Orchestration**. Four fields are on screen; two more sections are folded away underneath them.

1. **Title.** What it proves. This is the only required field.
2. **Projects.** Pick this *first*, and pick as many as the job needs — the field holds several. Until at least one project is in it, the Tests field has nothing to offer: it takes focus and then stays empty, which reads like a bug and is not one.
3. **Tests.** The tests that make up the job, listed under the project each one came from.
4. **Schedule.** Leave it on *On Demand* until you trust the suite.

Below those sit **Admin runtime settings** and **Advanced settings**. The first is reserved for Functionize administrators and is not something you need to open. The second is worth opening every time: it is where run order, re-run behavior and email alerts live — see Run Order, Re-runs and Alerts: An Orchestration's Advanced Settings.

The schedule deserves its own thought rather than a shrug, because it is the setting that decides what the suite costs. Scheduling an Orchestration covers all six cadences, including how to get a weekly job out of a product that has no Weekly option.



The orchestration then opens on its own summary: total runs, average time, how many tests it contains, and a Run History tab. **Run** starts the job and becomes **Stop** while it is in flight.

Build orchestrations by meaning, not by hand-picking

The single highest-leverage habit here is a consistent tag scheme on your tests — `smoke`, `checkout`, `slow`. Tags drive orchestration membership, so a well-tagged project lets you assemble a suite by what the tests *are* rather than by

remembering which ones to tick. A hand-picked list is correct the day you build it and quietly wrong a month later, when three new tests should have been in it and nobody noticed.

Tags are set in **General** → **Test details**. See Settings: Where Everything Lives and What to Change First.

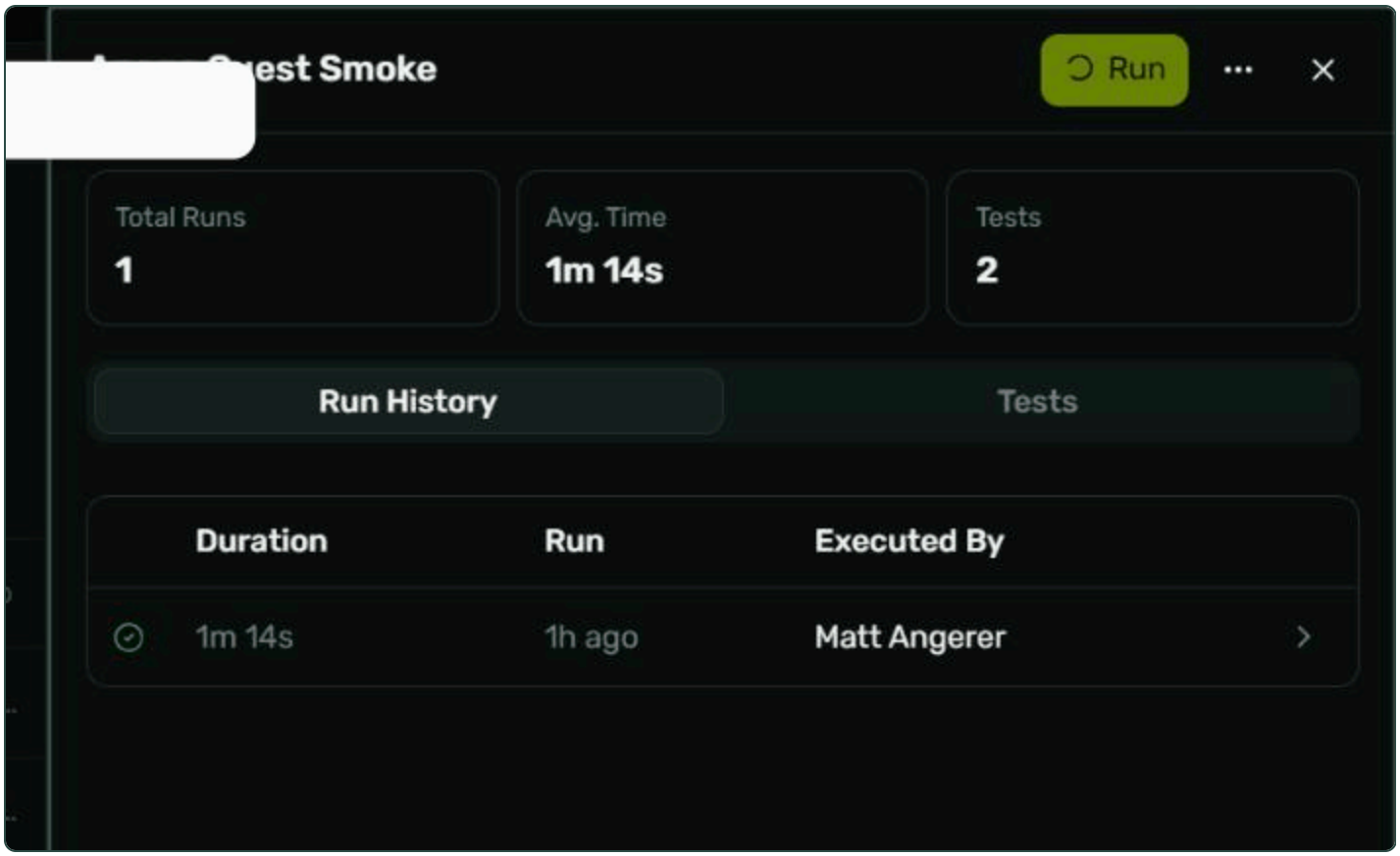
How many orchestrations should we have?

Usually three, and they correspond to how much patience the reader has:

Smoke	A handful of tests covering the journeys that must never break. Runs on every change. Minutes, not tens of minutes. If this is red, nothing else matters.
Core	The critical journeys. Runs on a schedule — nightly is common. This is the one that gates a release.
Full	Everything, including the slow and the peripheral. Once a week is usually enough — there is no *Weekly* option, so set the schedule to *Daily* and tick a single day. Nobody watches this one live; they read the trend.

The tiering matters more than the names. A single undifferentiated suite forces one answer to two different questions — "can we ship?" — and it answers neither well. There is more on tiering, and on what to do when a tier goes flaky, in Keeping a Suite Green: Flake, Tiers and Who Owns Red.

Reading the result



The history entry covers the whole job: result, duration, and who triggered it. Open any individual test from within it to see that test's own evidence — steps, browser pane and the Data tab — exactly as you would for a single run. See Understanding a Test Run and Its Results in Functionize Studio.

Across orchestrations and projects, Reports: Pass Rates, Trends and What They Actually Tell You is the view that answers whether things are getting better or worse.

Before you put one on a schedule

Three questions, and they are worth answering in writing:

- **Who reads a red result, and by when?** A named person, with a timescale. This single decision predicts whether the suite is still being read in six months better than anything else about it.
- **What does a failure block?** If the answer is "nothing", the orchestration is informational — which is fine, but say so, because a suite that gates nothing and is red for a week trains everyone to ignore it.
- **Is the suite stable enough?** A scheduled orchestration that is red half the time for reasons nobody trusts is worse than no orchestration. Fix the flake first.

Orchestrations and credits

An orchestration consumes credits for the tests it runs, the same as running them individually — the grouping is organizational, not a discount or a surcharge. What changes is *frequency*: a suite on a nightly schedule runs whether or not anyone changed anything, so the schedule is the thing to think about rather than the job. See How Credits Work.

Where to go next

- Exercise 6 - Group Tests Into an Orchestration — build one against the demo store in about ten minutes.
- Scheduling an Orchestration: The Six Cadences and What Each One Costs — the six cadences, and what each one costs.
- Run Order, Re-runs and Alerts: An Orchestration's Advanced Settings — parallel against sequential, re-runs and who gets the email.
- Managing Orchestrations: Filters, Bulk Actions and Run History — filters, bulk actions, the row menu and the run history.
- Execution Presets: Running One Suite Against Different Data — one suite, two data sets, no duplicated tests.
- Workflows: Scheduled Jobs Written in Plain Language — for scheduled work that is not simply "run these tests".
- Keeping a Suite Green: Flake, Tiers and Who Owns Red — how to keep a scheduled suite worth reading.
- Scaling From One Team to Many — the conventions to agree before several teams are building orchestrations side by side.