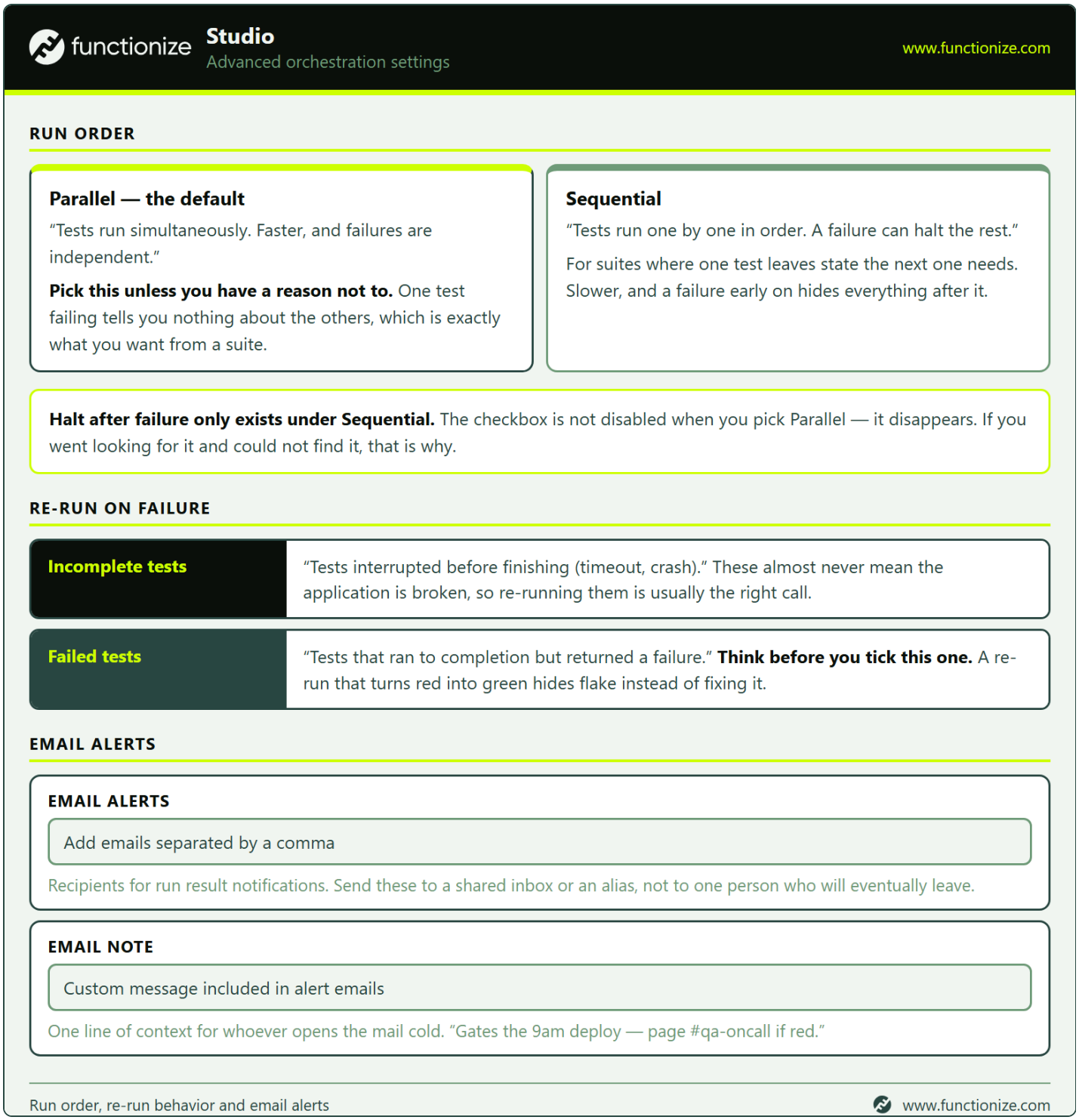
Run Order, Re-runs and Alerts: An Orchestration's Advanced Settings

Features · https://docs.functionize.com/hc/en-us/articles/43800801818519-Run-Order-Re-runs-and-Alerts-An-Orchestration-s-Advanced-Settings

The **New Orchestration** panel has four fields you cannot miss and two collapsed sections you easily can. **Advanced settings** is the second of them, and it holds four decisions that change what a suite does when something goes wrong.

None of them are required. All four are worth a deliberate answer once, per suite, rather than being left at whatever the default is because nobody opened the section.



Run order: parallel or sequential

Two cards, and Studio describes them in one line each.

Parallel (the default)	"*Tests run simultaneously. Faster, and failures are independent.*" A twenty-minute suite finishes in the time its slowest test takes. One test failing tells you nothing about the others, which is precisely what you want from a suite you are using as a health check.

Sequential	"*Tests run one by one in order. A failure can halt the rest.*" Slower by the sum of every test in the job, and a failure near the start can leave you with no information about anything after it.

Use Parallel unless you have a specific reason not to. The reason, when it exists, is almost always shared state: one test creates an account and a later one signs into it, or one test leaves a shopping cart in a condition the next test expects.

> **Sequential is a symptom worth treating**
> If a suite only passes in a fixed order, the tests are coupled, and coupled tests fail in ways that are hard to read — test 7 goes red because test 3 did something odd. Sequential is the right short-term answer and the wrong long-term one. Make each test set up what it needs and the coupling goes away.

Halt after failure, and why you cannot find it

Halt after failure is a checkbox that appears only when **Sequential** is selected. It is not grayed out under Parallel — it is not on screen at all. If you went looking for it and concluded it did not exist, this is why.

With it ticked, the first failure stops the job. The remaining tests are not run and not charged.

- **Tick it** when later tests genuinely depend on earlier ones, so running them after a failure produces noise rather than information.
- **Leave it clear** when the sequence is about resource contention rather than dependency — you want the whole picture, you just do not want the tests fighting each other.

Re-run on failure

Two checkboxes under the heading "*Configure test re-run behavior on failure.*" They look similar and they are not. Studio defines each one precisely, and the definitions are the whole argument.

Incomplete tests	"*Tests interrupted before finishing (timeout, crash).*" The test never reached a verdict. Something ran out of time or fell over. This says almost nothing about your application, so re-running is usually the right call.
Failed tests	"*Tests that ran to completion but returned a failure.*" The test got an answer and the answer was no. Re-running it asks the same question again and hopes for a different reply.

Turning on **Incomplete tests** is close to free judgment: it removes a class of red that was never about the application.

Turning on **Failed tests** is a real trade, and it is worth naming honestly. It will make your pass rate go up. It will not make your application better. A test that passes on the second attempt is a flaky test, and hiding that fact is how a suite quietly stops meaning anything. If you do turn it on, treat the fact that it *fired* as the signal — that test needs fixing, whatever color the orchestration ended up.

There is more on this trade in Keeping a Suite Green: Flake, Tiers and Who Owns Red.

Email alerts

Two fields at the bottom of the section.

Email Alerts	"*Email recipients for run result notifications.*" The field takes several, separated by a comma.
Email Note	"*Custom message included in alert emails.*" One line of your own, carried in every alert this orchestration sends.

Two habits make these worth having:

- **Send to a shared address, not a person.** An alias or a team inbox survives somebody changing role. A personal address turns into a silently unread mailbox the week they move teams.
- **Use the note to say what to do.** The alert already carries what happened. What the reader needs is what it means and who acts. "*Gates the 9am deploy. If this is red, hold the release and page the on-call QE.*" is a better note than "*Nightly regression.*"

An alert with no stated owner and no stated consequence becomes a filter rule within a month. See Keeping a Suite Green: Flake, Tiers and Who Owns Red for the wider version of this argument.

Admin runtime settings

The other collapsed section, sitting just above Advanced settings, is **Admin runtime settings**. It pins the execution runtime an orchestration uses and is reserved for Functionize administrators — there is nothing in it a customer needs to set, and the defaults are correct for every normal suite. Leave it closed.

A sensible starting configuration

For a first orchestration, and for most of the ones after it:

- Run order: **Parallel**.
- Re-run **Incomplete tests**: on.
- Re-run **Failed tests**: off.
- Email alerts: a team alias, with a note naming what the suite gates.

That configuration is fast, does not lie to you about flake, and reaches somebody who can act. Change it when you have a reason you could explain out loud.

Where to go next

- Orchestrations: Running a Suite as One Job — the concept and the four main fields.
- *Scheduling an Orchestration* — the six cadences, and what each one costs.
- Managing Orchestrations: Filters, Bulk Actions and Run History — how to edit these settings on a suite that already exists.
- Keeping a Suite Green: Flake, Tiers and Who Owns Red — what to do when the re-run checkbox starts looking tempting.