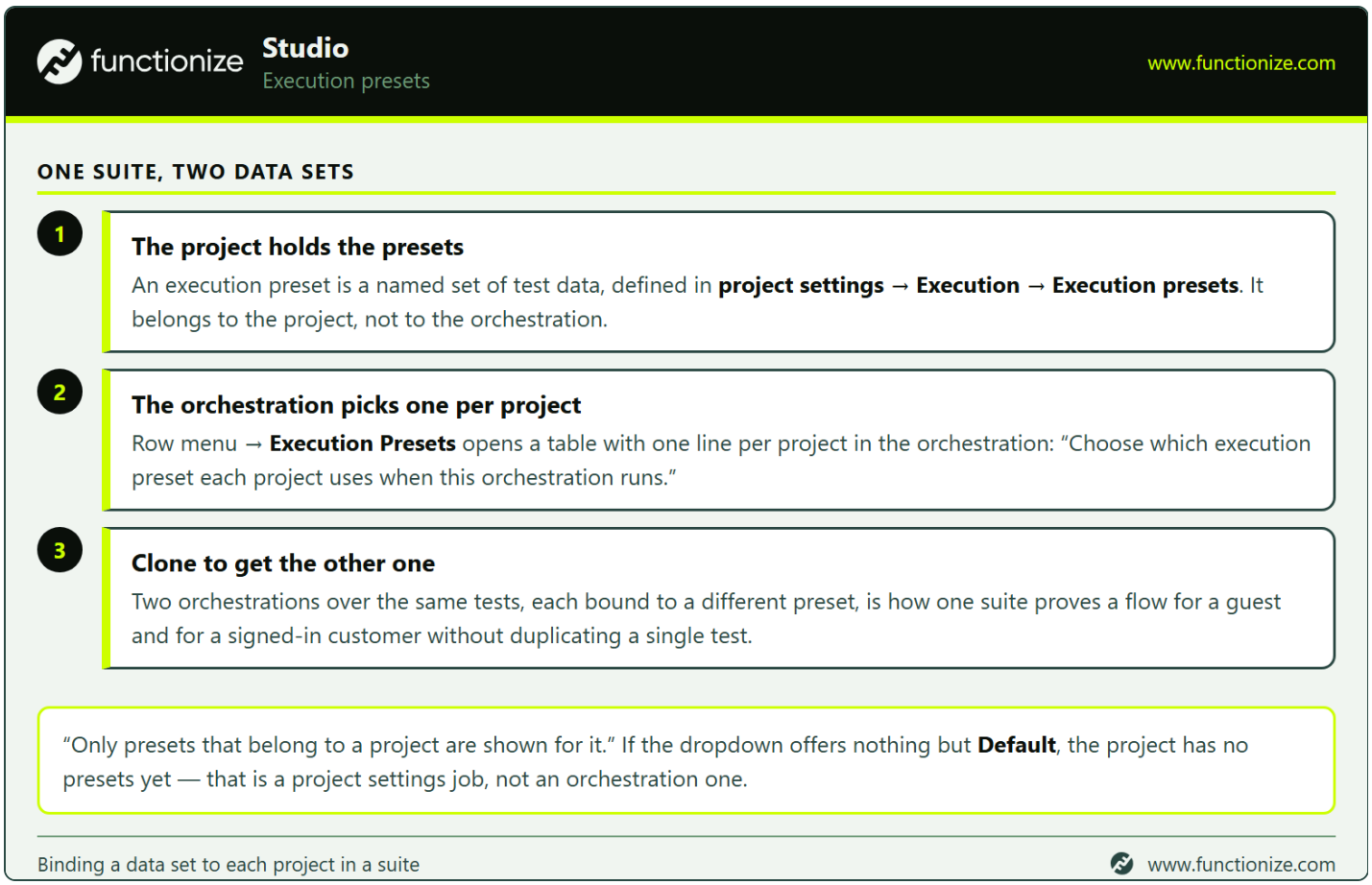
Execution Presets: Running One Suite Against Different Data

Features · https://docs.functionize.com/hc/en-us/articles/43800795106839-Execution-Presets-Running-One-Suite-Against-Different-Data

Most teams reach the same wall at about the thirtieth test: the same journey needs proving twice. Once as a guest and once signed in. Once against staging and once against a UAT environment. Once with a US address and once with a German one.

The obvious move is to duplicate the tests. The better move is to leave the tests alone and change the data underneath them, and that is what an orchestration's **Execution Presets** are for.



What an execution preset is

Studio's own definition is the clearest one, and it is printed at the top of the section:

> "Execution presets are reusable variable sets that configure how tests run in this project. Select a preset when running tests or starting sessions to apply its variables automatically."

Three things follow from that sentence. A preset is a set of **variables**. It belongs to a **project** — you define presets in project settings under **Execution → Execution presets**, and the section only appears on a project, never on an individual test. And it applies **when running tests or starting sessions**, which is broader than most people assume.

Creating one asks for a single thing, a **Preset Name**. You then open it and fill it with variables — the drawer is headed "*Execution preset variables*" and names the preset it belongs to.

A preset might hold a base URL, a set of credentials, a currency, a customer reference — whatever your tests read rather than hard-code. There is more on building them in Test Data: Execution Presets, Variables and Secrets.

What the orchestration adds

Because a preset is defined once and chosen many times, there are three places it gets applied, and knowing all three is what makes the feature useful rather than decorative.

One test run	**Run Test options → Run with preset**, in the editor or from the Tests screen's row menu. Studio asks "*Choose which execution preset to apply for this run.*" Nothing about the test changes — only this run.
A whole orchestration	Row menu → **Execution Presets**, one line per project in the suite. Covered below.
A session	Chosen when the session starts, so the agent builds and tries things against the data you meant.

The orchestration does not define presets. It *chooses* one, per project, for the duration of its own runs.

You can reach a project's presets without opening the project at all: the **Projects** list has **Execution Presets** on every row's menu, next to **Settings**.

Open the row menu on any orchestration and choose **Execution Presets**. Studio opens a table with one line for every project in the suite, and states the rule at the top:

> "Choose which execution preset each project uses when this orchestration runs. Only presets that belong to a project are shown for it."

Each line is a dropdown. **Default** is always there; anything else is a preset somebody built on that project.

The pattern this unlocks

Two orchestrations over exactly the same tests, each bound to a different preset.

Guest checkout — anonymous	The tests, with the preset that carries no credentials.
Guest checkout — signed in	The same tests, cloned, with the preset that signs in first.

Nothing is duplicated except the grouping. Fix a test once and both suites improve. Add a test to the project and it goes into both. This is the difference between a suite that scales and one that doubles in size every time somebody asks a reasonable question.

Clone is the mechanism. Build the first orchestration, clone it, rename the clone, and change one dropdown. That is the whole workflow.

> **Put the preset in the title**
> Two orchestrations with identical tests and different data are indistinguishable in the list unless the name says so. *Checkout — UAT data* and *Checkout — prod-like data* cost nothing to type and save somebody a confusing ten minutes later.

When the dropdown offers nothing but Default

That means the project has no presets yet. It is not an orchestration problem and there is nothing to fix on this screen — go and build a preset in the project's settings first, then come back. The list here is populated from the project, every time.

The same is true in reverse: a preset you add to a project later shows up in this dropdown without you touching the orchestration.

A multi-project suite has one line per project

Because the **Projects** field on an orchestration takes more than one project, a suite can span several — and each project brings its own presets. The table gives you a line for each, so a three-project release check can run each project against the data that project needs.

This is powerful and it is also the point at which a suite becomes hard to reason about. If you find yourself setting three different presets on one orchestration, it is worth asking whether it should be three orchestrations that a pipeline triggers together. See Scaling From One Team to Many.

What this does not do

- **It does not change the tests.** A test that hard-codes a value ignores every preset you point at it. Presets only help tests that were written to read their data.
- **It does not apply to a single manual run.** Running a test on its own uses the test's and project's own settings. The preset chosen here applies when *this orchestration* runs.
- **It is not an environment switch by itself.** It can carry the base URL that makes it one, but only if your tests take the URL from data rather than from the step.

Where to go next

- Test Data: Execution Presets, Variables and Secrets — building the presets this page selects from.
- Managing Orchestrations: Filters, Bulk Actions and Run History — where the Execution Presets item lives, and what Clone does.
- Settings: Where Everything Lives and What to Change First — the project settings map, including the Execution group.
- Scaling From One Team to Many — conventions to agree before several teams are doing this side by side.