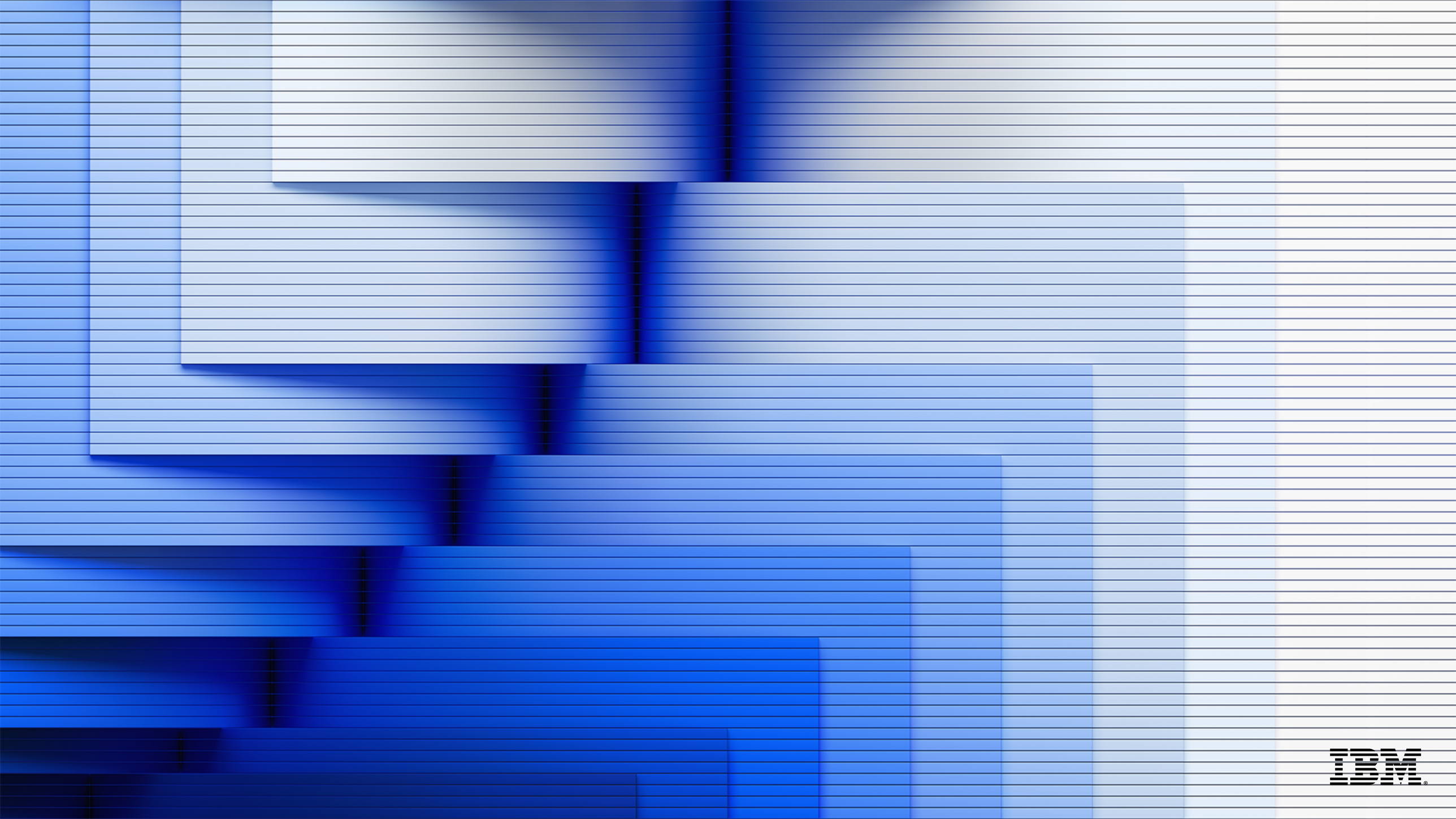


Science in the agentic era: structured experimentation with *ado*

Alessandro Pomponio
Research Software Engineer

IBM

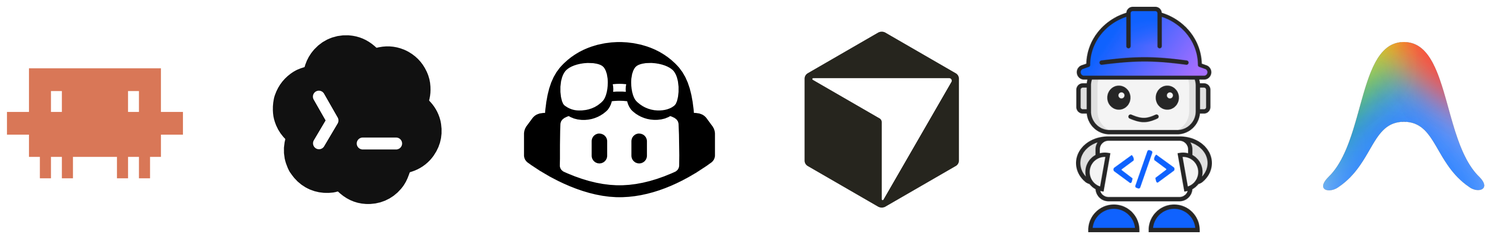
What is a coding agent?

What is a coding agent?

coding agent = llm + harness

What is a coding agent?

coding agent = llm + harness



What do coding agents do?

Explore

What do coding agents do?

Explore, Plan

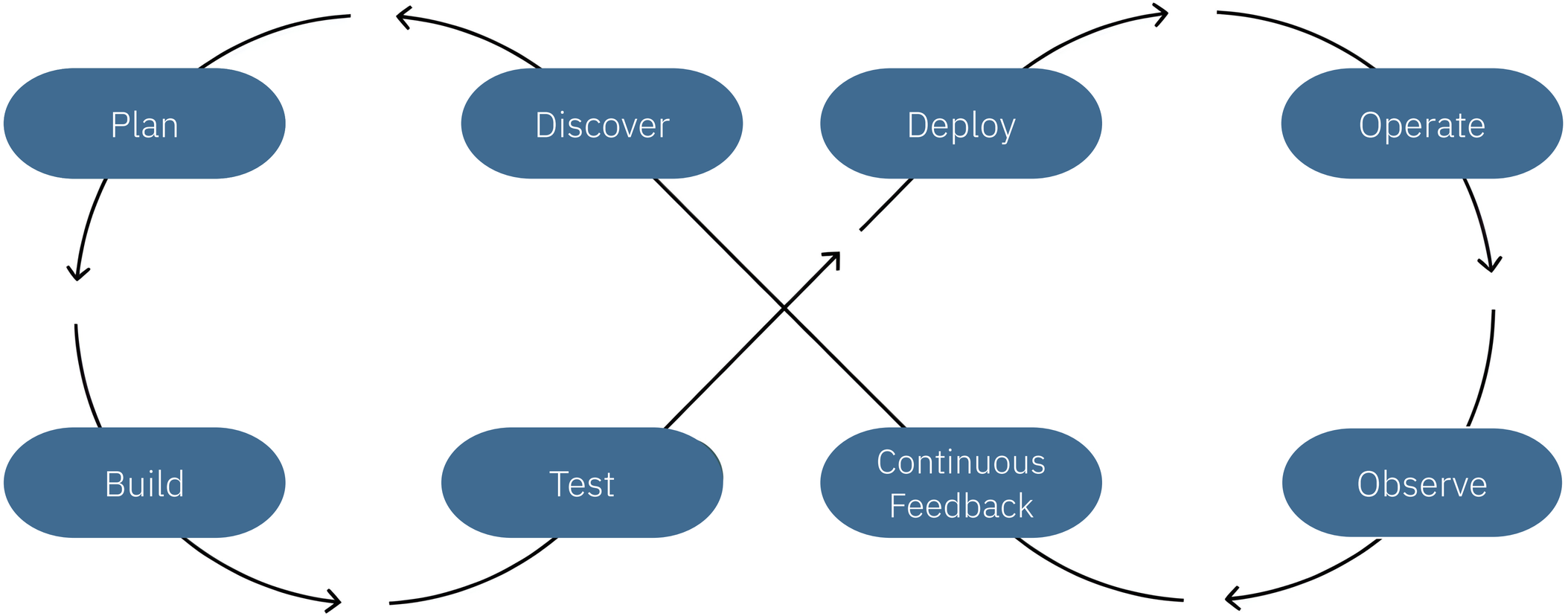
What do coding agents do?

Explore, Plan, Execute

What do coding agents do?

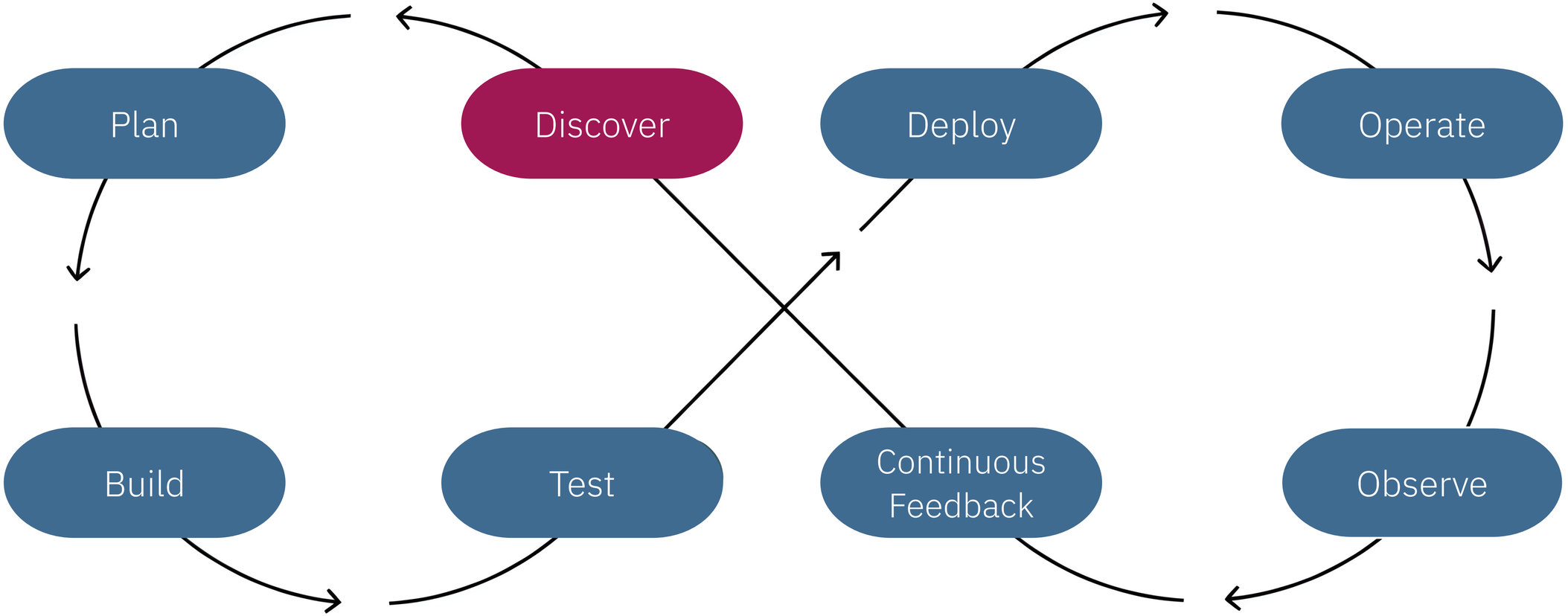
Explore, Plan, Execute, Verify

DevOps



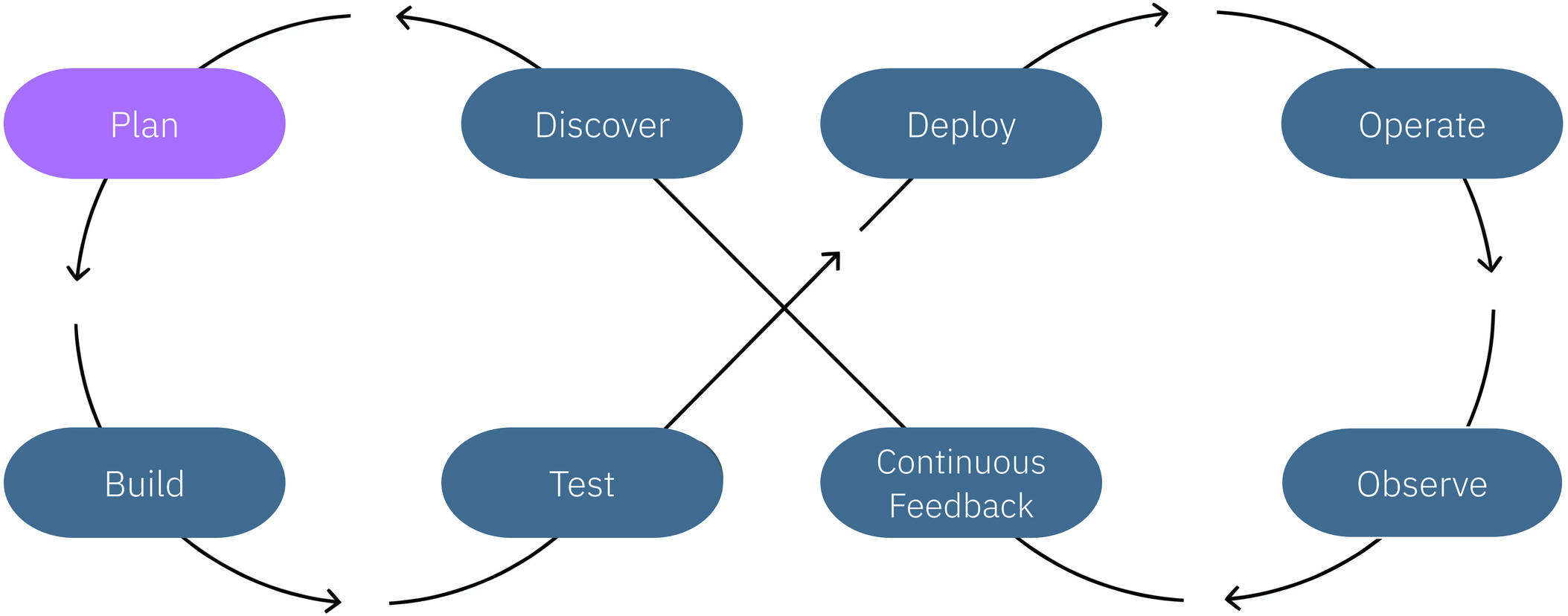
Credits: CircleCI; Atlassian

DevOps



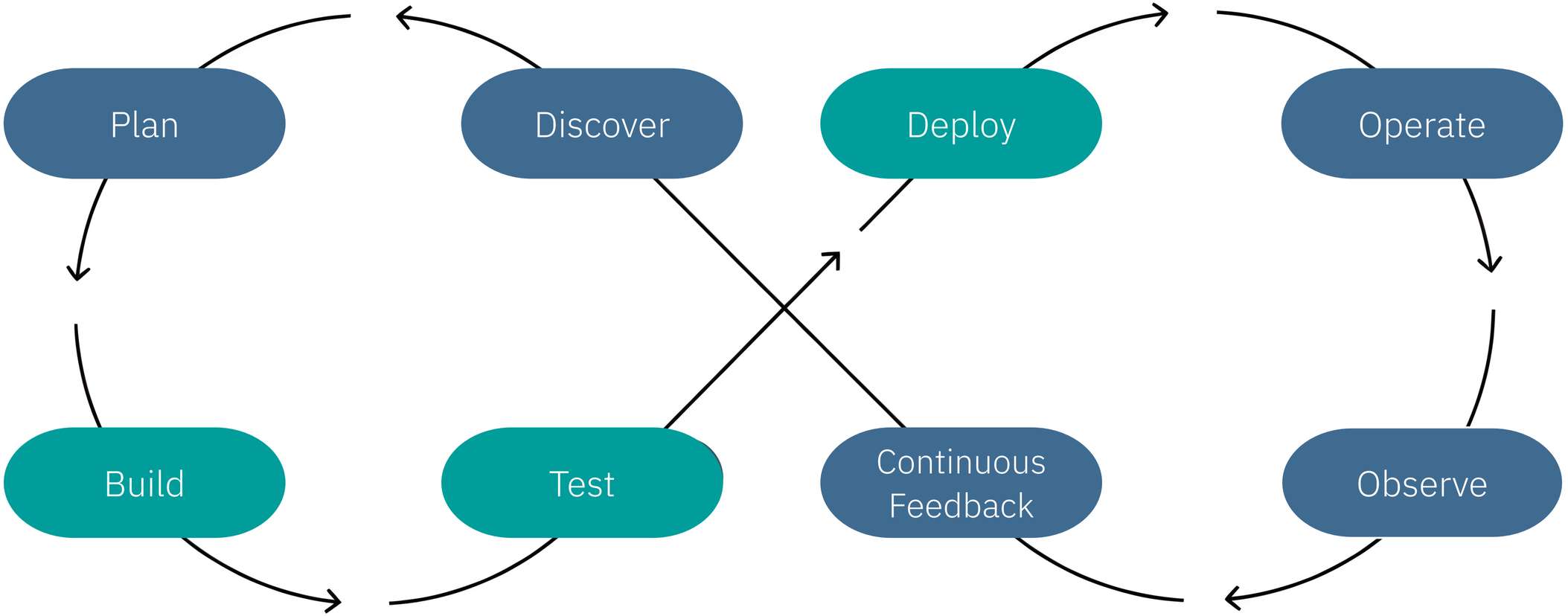
Credits: CircleCI; Atlassian

DevOps



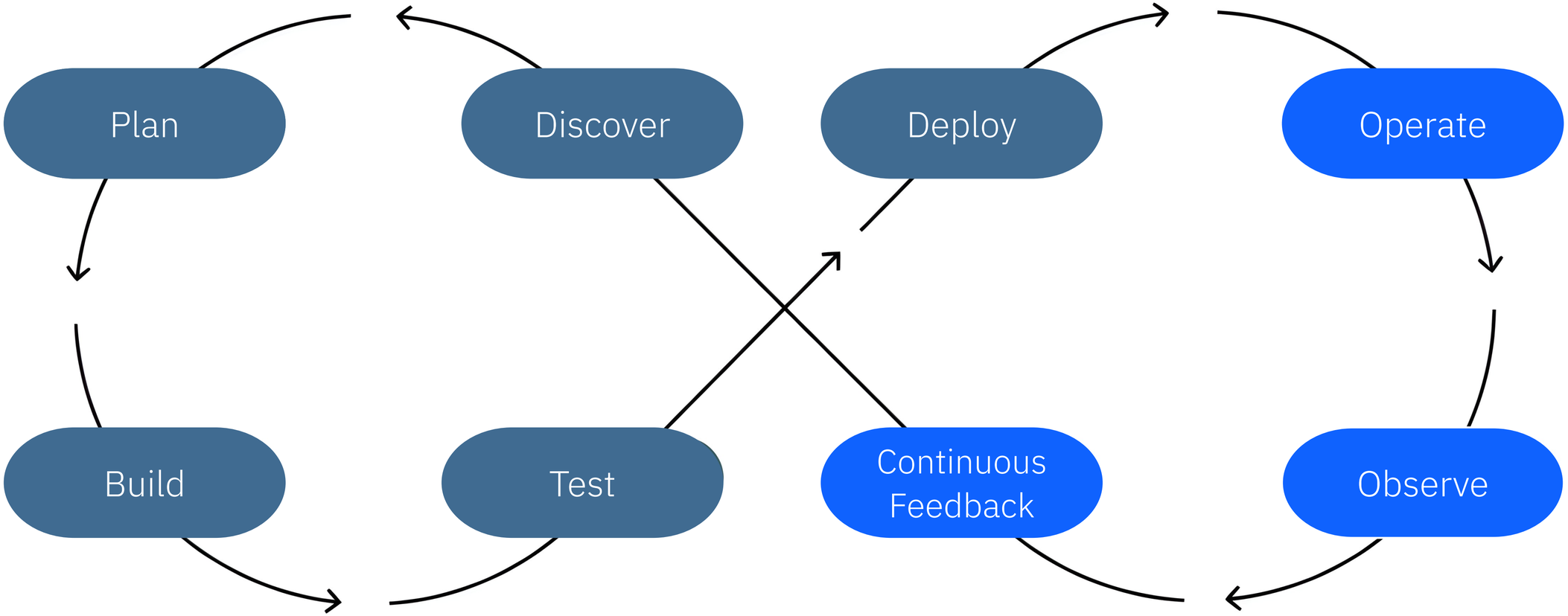
Credits: CircleCI; Atlassian

DevOps



Credits: CircleCI; Atlassian

DevOps



Credits: CircleCI; Atlassian

Tools turn LLMs into agents

Explore

Agents can use tools to gather contextual information from various sources, like git repositories, external logging systems, etc.

Plan

They can use this information to create structured and grounded plans that identify intermediate steps, risks and validation mechanisms.

Execute

The harness provides agents with (relatively) isolated environments where they can read and write from the filesystem, execute commands, and call APIs.

Verify

Agents can verify the changes by using language servers, linters, tests, and monitoring instrumentation.

Ref: Ning et al. Code as Agent Harness

What do scientists do?

Explore

What do scientists do?

Explore, Plan

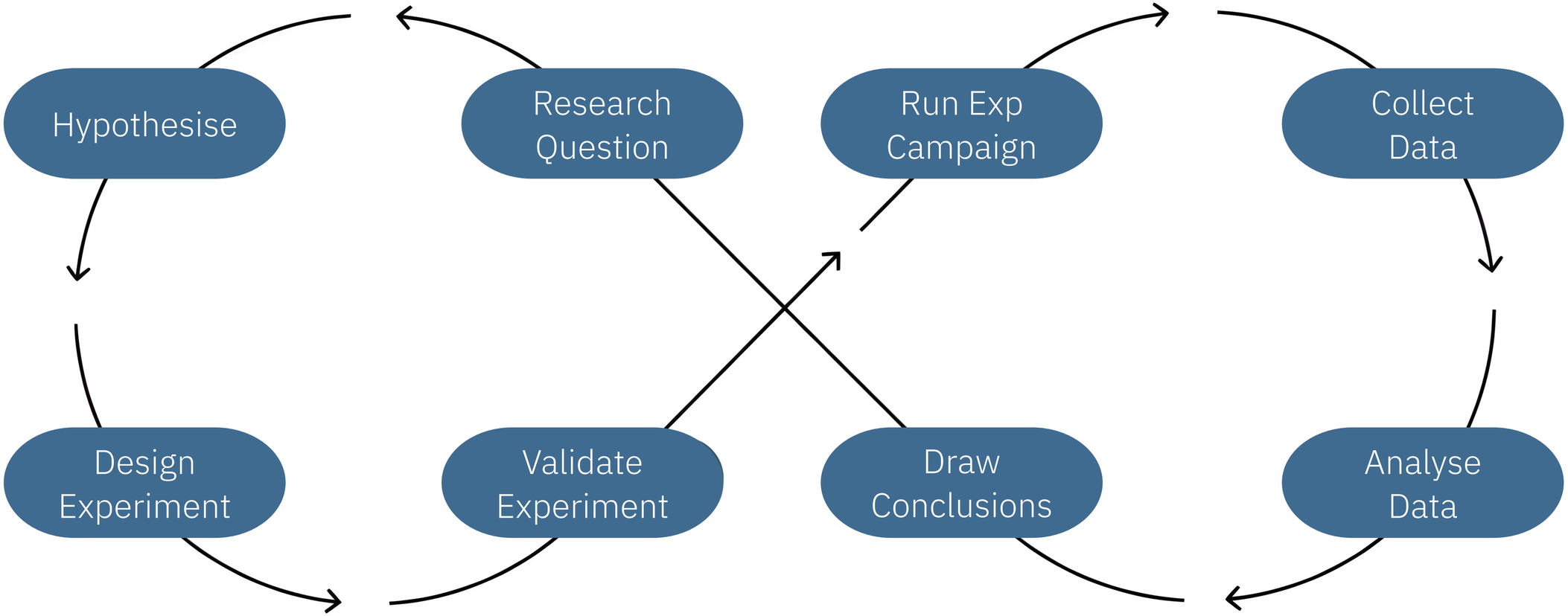
What do scientists do?

Explore, Plan, Execute

What do scientists do?

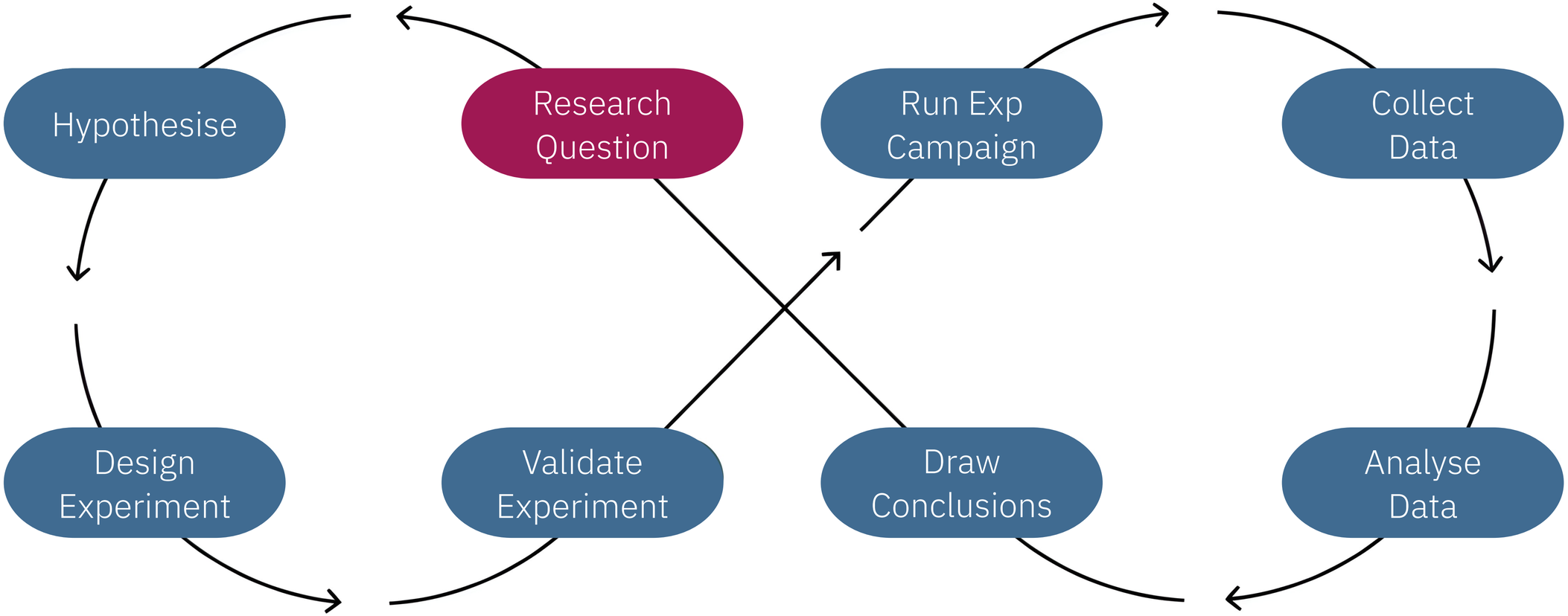
Explore, Plan, Execute, Verify

The Scientific Method



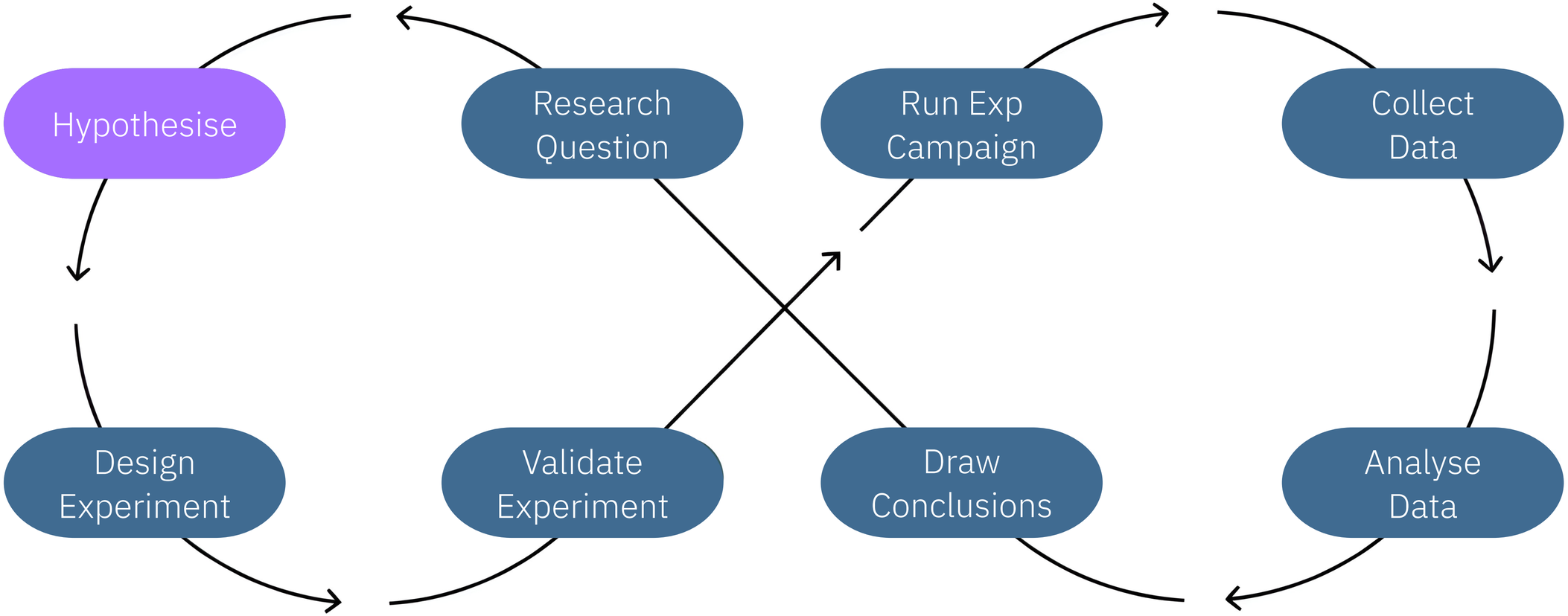
Credits: CircleCI

The Scientific Method



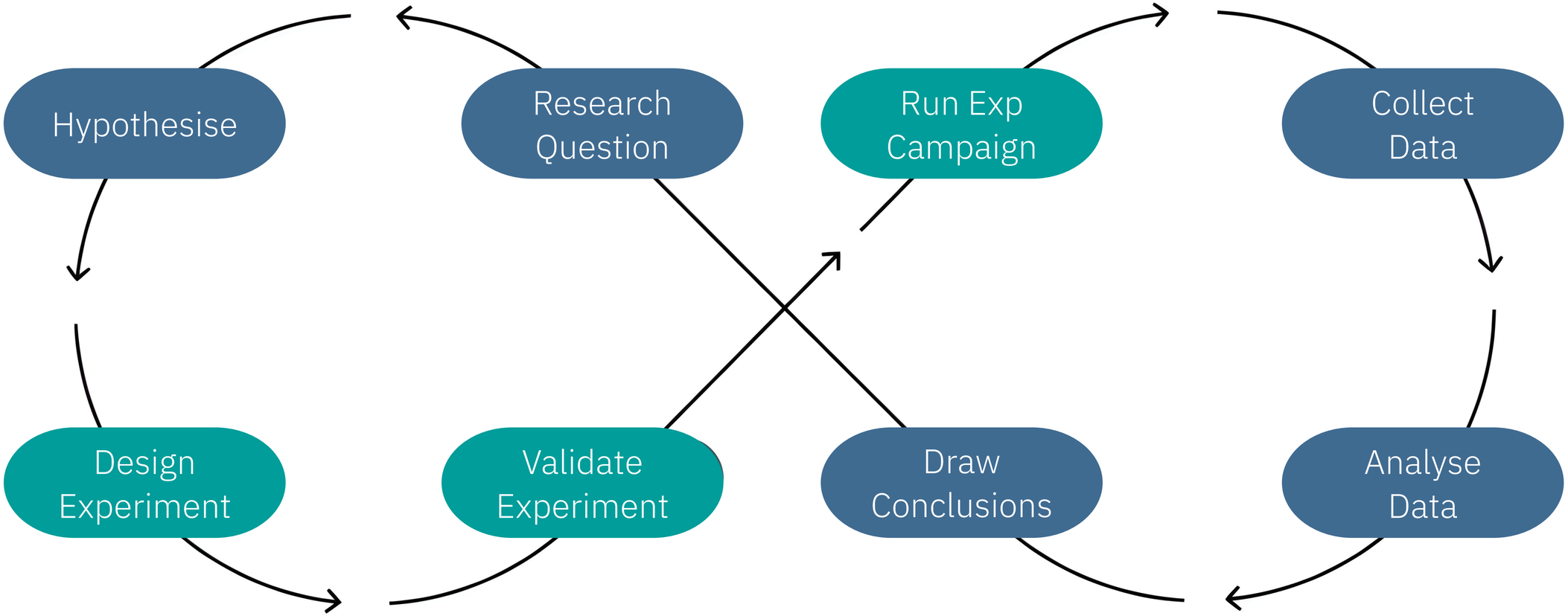
Credits: CircleCI

The Scientific Method



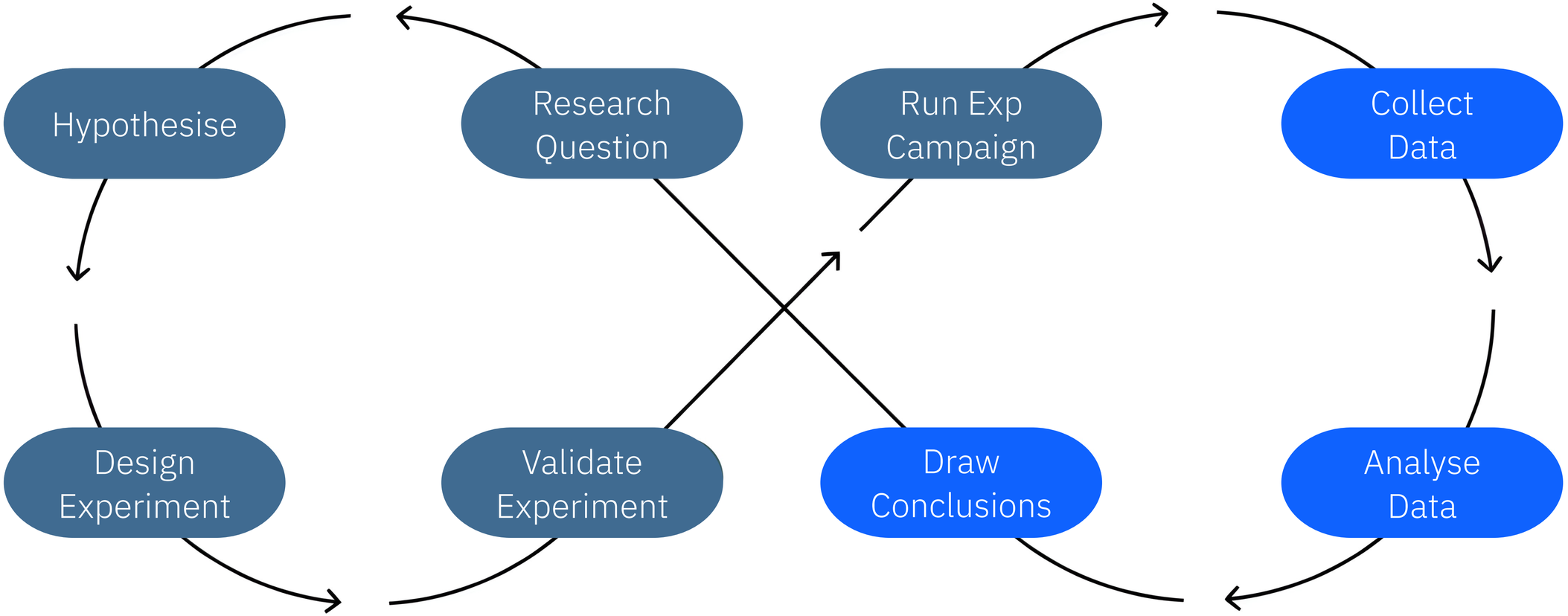
Credits: CircleCI

The Scientific Method



Credits: CircleCI

The Scientific Method



Credits: CircleCI

What’s missing for agents in science?

science agent = coding agent + …

Our vision: *ado*

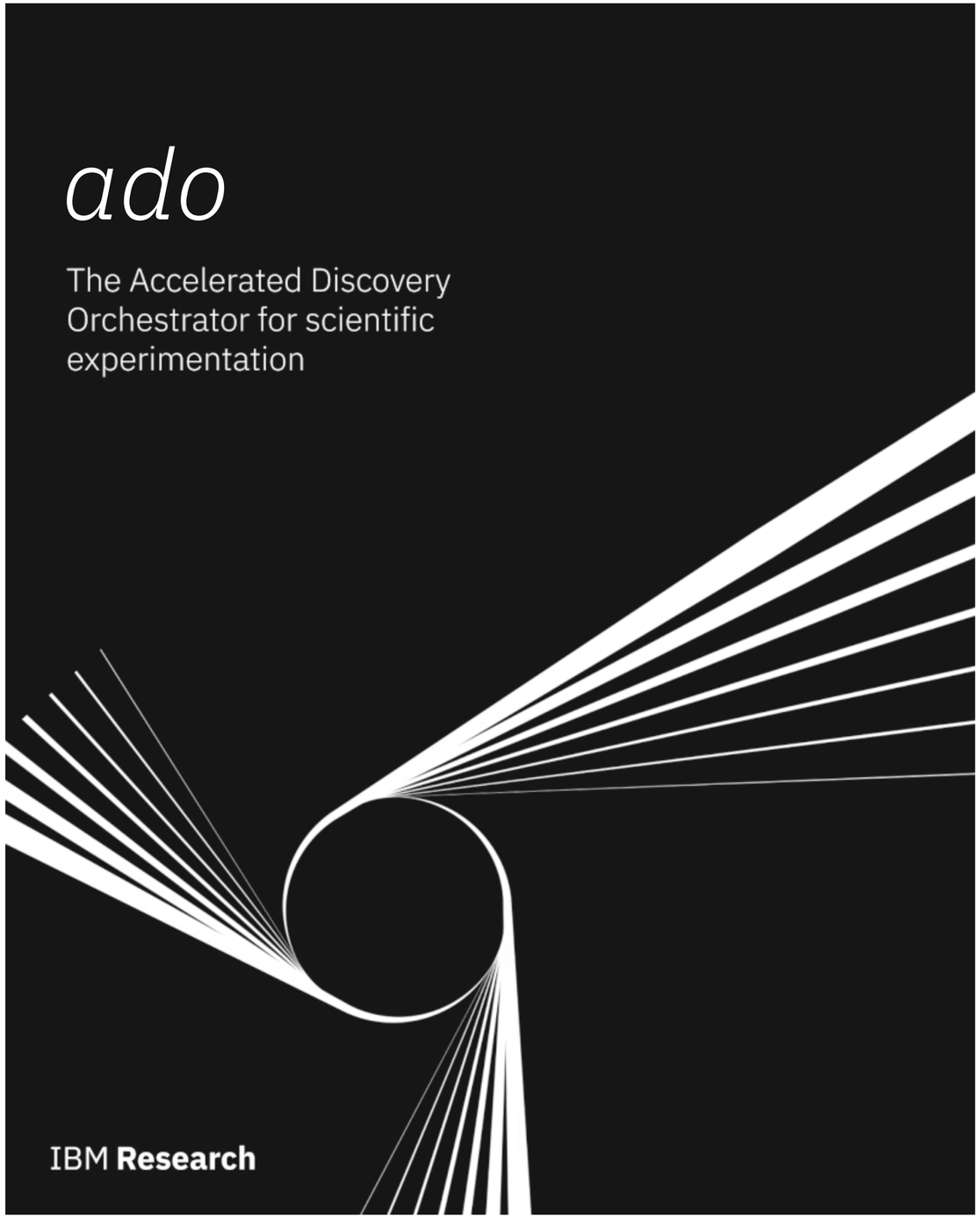
ado is the toolbox for agents in science:

- Schema-based approach to experimentation.
- Built-in provenance tracking.
- Automated results storage with support for collaborative projects.
- Extensible core architecture.
- Scalable execution and intuitive CLI.
- *And more!*

Repo: github.com/IBM/ado

Docs: ibm.github.io/ado/

Paper: doi.org/10.21105/joss.10304



Our vision: *ado*

science agent = coding agent + ado

ado as your science toolkit

Explore

ado provides methods to store and retrieve experiment plans, configurations, and results from a persistent and queryable data store.

Plan

Agents can use this information to analyse results, discover trends in the data, and find gaps. Plans and reports are written to the data store to track progress.

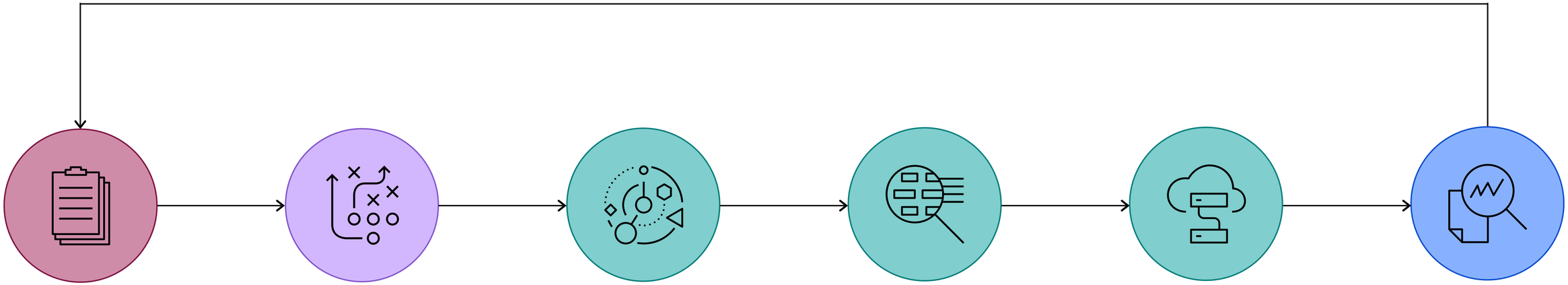
Execute

ado allows agents to launch experiment campaigns, while its orchestration layer automatically tracks provenance information and sampling trajectories.

Verify

All resources follow strict schemas which are validated on creation. Additionally, agents can perform advanced troubleshooting by inspecting execution logs and execution traces, agents.

The *ado* loop



Retrieve existing plans, experiment configurations, and measurements

Create a plan for the next steps in your research

Define the parameter space you want to explore and the experiments you want to run

Run the experiments on the points of the parameter space

Store results in shared persistent and queryable storage

Analyse the results of the exploration and use them to draw conclusions

“And they lived happily ever after.”

git gud

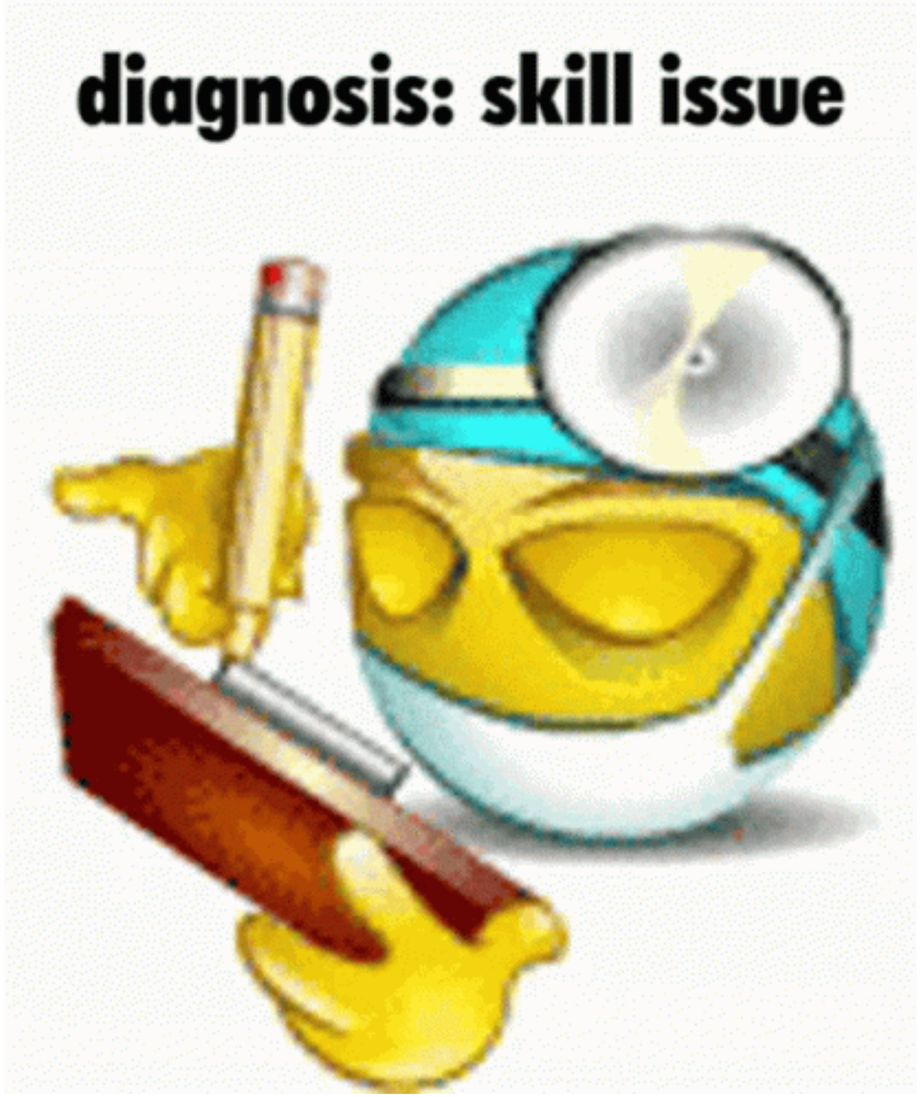
Coding agents work best on established frameworks. When dealing with new ones they haven't been trained on, they often:

- ***Don't know where to look*** during their exploration step.
- ***Hallucinate*** commands, classes, methods, ...
- ***Take a long time and burn through tokens*** to produce something that works.
- ***End up with "creative solutions"*** trying to circumvent problems caused by poor exploration.
- ***Create many single-use scripts*** that are impossible to keep track of.

git gud

Coding agents work best on established frameworks. When dealing with new ones they haven't been trained on, they often:

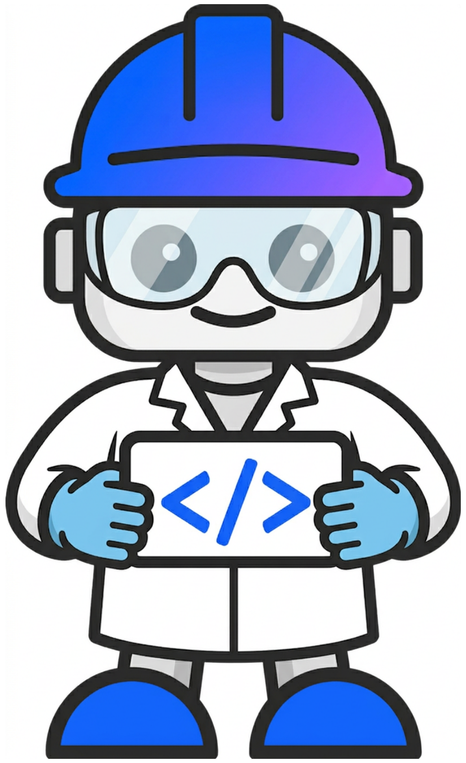
- ***Don't know where to look*** during their exploration step.
- ***Hallucinate*** commands, classes, methods, ...
- ***Take a long time and burn through tokens*** to produce something that works.
- ***End up with "creative solutions"*** trying to circumvent problems caused by poor exploration.
- ***Create many single-use scripts*** that are impossible to keep track of.



Upskilling agents

Agent skills can work wonders, turning an expert's domain knowledge into grounded agent behaviour:

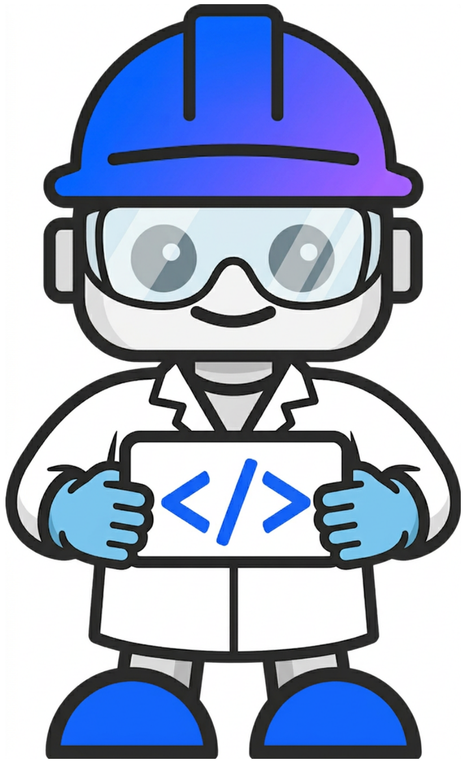
- ***Prevent agents from guessing*** and reduce hallucinations.
- ***Save agents from going down the wrong path*** with clear dos and don'ts.
- ***Provide step-by-step workflows*** with decision branches and stopping conditions to improve reliability.



Upskilling agents

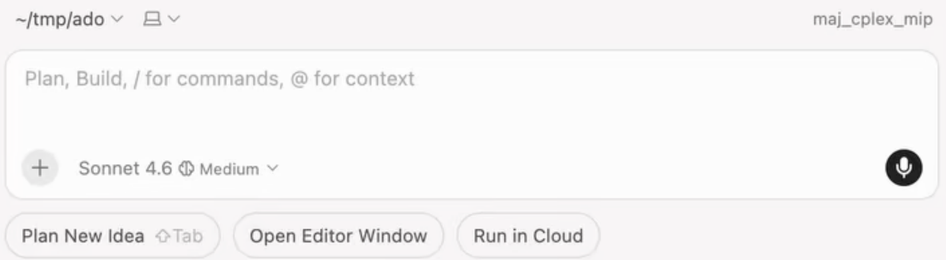
Agent skills can work wonders, turning an expert's domain knowledge into grounded agent behaviour:

- ***query-ado-data***: how to efficiently query our data store to find existing plans, similar parameter spaces, and more.
- ***formulate-discovery-problem***: example-driven methodology on how to choose appropriate experiments, defining the search spaces, and the experiment campaigns.
- ***conduct-empirical-study***: step-by-step workflow to conduct empirical studies with *ado*. Includes the full explore, plan, execute, and verify loop.



Demo time

Catching up on the status of a research project


~/tmp/ado
maj_cplex_mip
Plan, Build, / for commands, @ for context
Sonnet 4.6 Medium
Plan New Idea Tab
Open Editor Window
Run in Cloud

Investigating failing experiment campaigns

- Mar: three production 60-min spaces (bab6, sorrell3, atlanta-ip) each receiving 200-sample LHU runs
- Apr 1: 9 timeseries ranking operations covering 3 metrics × 3 problems
- Apr 2–3: bab6 pruned TPE space + 200-sample Optuna run — **completed successfully**
- Apr 16–17: sorrell3 pruned 20-min TPE space created + Optuna run — **FAILED after 6 trials**

Suggested next steps

1. Fix the failed sorrell3 TPE operation (highest priority)

The operation `cplex_sorrell3_tpe_mip_gap_20m` failed because all 6 submitted trials reported `NaN` for `mip_gaps-mean`. The most likely cause is that `sorrell3.mps.gz` (the compressed filename used in the new space) was not accessible on the remote cluster. Check:

- What filename the successful sorrell3 LHU operations used (`raytune-1.6.1.dev35+...lhu_sampler-f65a1e`): run `uv run ado get operation raytune-1.6.1.dev35+c0888ef.dirty-lhu_sampler-f65a1e -o yaml` and inspect `mps_file`
- If the LHU run used `sorrell3.mps` (uncompressed), update the new TPE space's `mps_file` value to match and re-run the 200-sample Optuna operation

2. Create and run a TPE expansion for `atlanta-ip`

bab6 and sorrell3 both now have pruned TPE spaces; atlanta-ip does not. The timeseries ranking data for atlanta-ip (3 operations covering mip_gap, objective, best_bound) already exists and should be used to:

1. Identify which parameter levels to prune (same analysis as for bab6/sorrell3)
2. Create `atlanta_ip_tpe_expanded_60m_gap_pruned` with appropriate dimensions and a 60-min time limit (use the bab6 TPE space YAML as a template)
3. Run a 200-sample Optuna TPE operation

3. Analyse the bab6 TPE results

The 200-sample bab6 Optuna run completed on Apr 3. A report was written (`raytune-1.7.1.dev58+d8be1ae.dirty-optuna-72f5e5_2026-04-03_report.md`) — consider:

- Identifying the top-k CPLEX configurations from TPE and comparing their speed factors vs the LHU-derived ranking
- Running a final timeseries ranking (`timeseries_ranking`) on `space-6aeea0-default` using the TPE-found configurations as the entity set

4. Cross-problem TPE synthesis

Once sorrell3 (fixed) and atlanta-ip TPE runs are complete, a synthesis analogous to `timeseries_ranking_bab_sorrell_atlanta_2026-04-01_synthesis.md` would be valuable: do the top configurations found by TPE on each problem share common parameter patterns? This would point toward a "universal" CPLEX configuration or confirm that optimal tuning is strongly problem-specific.

Running diagnostic experiments

```
```text
Total entities sampled:                                  6
```

Re-lint the report uv, tail

Completed 7 of 7 Write report

Report written to `reports/cplex_mip/raytune-1.7.1.dev101+g1b03dc2de.d20260416185741-optuna-fe1713_2026-04-24_report.md`.

Summary

Operation: `cplex_sorrell3_tpe_mip_gap_20m` — 200-trial Optuna TPE search on the pruned `sorrell3_tpe_expanded_20m_gap_pruned` space, targeting `mip_gaps-mean` with a 20-min per-trial time limit.

What happened: Total failure. Only 6 of 200 trials were submitted (cluster had ≤ 6 free slots at the time), the operation ran for just **2 min 44 s**, and every trial returned `NaN`. No ado measurements were written at all. RayTune then crashed trying to call `get_best_result()` with no valid results.

The 2.7-minute total runtime is the key indicator — with a 20-minute per-trial time limit, trials cannot have completed normally. The experiments failed during **initialisation**, before CPLEX started solving.

Most likely root causes (in priority order):

1. **Operator/plugin change between `dev58` (bab6 success, Apr 2) and `dev101` (this run, Apr 16)** — both builds from source. A bug introduced in `plugins/operators/ray_tune` or `plugins/custom_experiments/cplex_mip` between those commits could cause every trial to throw at startup. Check with:

   ```
   git log --oneline d8be1ae..g1b03dc2de -- plugins/custom_experiments/cplex_mip/ plugins/operators/
   ```

2. **`sorrell3.mps.gz` not staged on the cluster** — the file exists locally and is in `additionalFiles`, but if the port-forward or upload failed the file wouldn't be present and every `model.read()` call would fail immediately.

Recommended next action: Run a 1-sample diagnostic operation with `wait: true` to capture live cluster logs and identify the exact exception before re-running the full 200-trial search.

Review +326981 -55 | Commit & Push

Run the diagnos

Sonnet 4.6 Medium
   ```

Launching experiment campaigns

Check operation details and result measurements uv, 2+

Thought for 3s

Extract meaningful columns from results CSV uv, 3+

Export results to file and find mip_gap columns uv, 2+

Print all non-empty result columns python3

Thought for 3s

Show entity measurements including target metrics uv, 2+

Thought for 7s

The diagnostic completed perfectly. Here's what we now know:

Diagnostic results — 1 trial, warm-start point 1:

Metric	Value
`solve_times`	1200s × 5 seeds (time limit hit, as expected)
`objective_values`	−16.0 across all 5 seeds
`mip_gaps`	9.1%, 14.2%, 14.9%, 15.1%, 16.1%
`mip_gaps-mean`	**~13.9%**
`solve_statuses`	`time limit exceeded` × 5 (correct)

Root cause confirmed: The April 16 failure (`dev101` at commit `g1b03dc2de`) was a **code regression in the ray_tune operator or cplex_mip plugin** that caused all trials to return NaN. The current code (`dev131` at `g20dca7e18`) is completely healthy — all 5 seeds ran to the time limit and returned valid measurements.

The full 200-trial TPE operation (`operation_tpe_mip_gap.yaml`) can be re-submitted now. The only thing to note is that the space already has 1 measurement from this diagnostic trial, so the warm-start for that entity is already covered. Do you want to go ahead and submit the full operation?

1 Review +327096 -57 Commit & Push

Send follow-up Sonnet 4.6 Medium

conclude-presentation/SKILL.md

We’re eager to collaborate and to hear from you!

- Try the project: tell us what work and what doesn’t.
- Share your use case and the problems you’re trying to solve.
- Open an issue if something is missing or unclear.
- Contributions are welcome and appreciated!

Give us a star! ⭐

Get in touch!

Thank you