How Faros Accelerates Your Organization's Transformation to Data-Driven Engineering

This document provides an overview of how Faros provides a future-ready alternative to a 100% build-it-yourself approach to engineering productivity metrics.

Benefits of Faros to Your Organization

Faros provides a **buy**-*and*-**build** solution that provides you with the following benefits:

- **Shorter Lead Time:** Get insights faster, with Faros taking on an estimated 80 percent of the undifferentiated data collection, normalization, and analysis. With Faros in place, your enablement team will be able to turn its focus and precious time to taking action based on the data instead of spending many cycles on collecting and structuring it. Faros will also help you work around tricky subjects and avoid red herrings like Jira standardization in the context of a holistic view of engineering productivity.
- **Lower Cost:** Reduce the internal effort to become industry experts in the domain of engineering productivity. Leveraging Faros's expertise and dedication to this topic, you can learn what to measure, what good looks like, and what each role needs to pay attention to and act upon. We can assist with the transformation aspects as well, including alignment, communication, and rollout.
- **Reduce Risk:** Make sure you are measuring the right things in a holistic manner against industry best practices. Faros works with other huge organizations like Salesforce, Box, and Autodesk and is in frequent and close contact with the industry experts and analysts. You'll also avoid expensive mistakes, e.g. designing a solution that doesn't scale or doesn't lend itself to AI/ML.
- **Future Proof Your Metrics:** Companies like yours never stand still. M&A, consolidation, new business needs, new visibility requirements… these will all impact what you'll want to measure, analyze, and see. Faros is optimally architected to take those changes in stride. Internal efforts frequently need to return to the drawing board every time a change impacts your environment.

What about the sunk cost?

We realize you've already invested many years of effort to get as far as you've come. One Faros customer estimated investing $4M upfront cost to build their own DORA metrics, a "quagmire" initiative that eventually got defunded.

Your effort to-date will not be lost if you decide to use Faros. **You don't have to throw away anything you've done.** Everything you've invested thus far can be reused within Faros. Faros will help you build on top of that, marrying your existing metrics with new focus areas you have not been able to tackle yet due to limited resources.

What will you gain? Expertise + Platform + Shortcuts

Partnering with Faros, you will be turbo-charging your engineering productivity capabilities:

1. **You will be gaining a team of engineering productivity experts that will act as an extension to your internal team.** Today, we estimate you are spending many internal cycles to acquire the knowledge and expertise to do engineering productivity properly. The Faros team will become an extension of your internal teams, contributing its expertise in engineering productivity to be combined with your context-specific knowledge of your needs, challenges, organization structure, strategy and objectives.

2. **You will be able to generate new business insights faster**, because so much of the foundation has already been built by Faros, and Faros does so much of the heavy lifting. For example, the Faros team is currently training LLMs to give the right answers in the context of engineering productivity metrics to natural language queries. You'll be able to reinvest your sizable team's time to uncover insights in new areas and enact change within the company as part of your transformation to data-driven transformation engineering.

3. **You will benefit from innovation and advances in leading best-of-breed open source technologies** that are packaged within Faros, including Metabase (BI layer), Hasura (GraphQL), Airbyte (connectors to data sources), and n8n (automation).

What's in this document

This document describes how Faros handles many of the data integration and analysis challenges most large scale engineering organizations face:

Cross-Tool, Cross-Team Data Normalization

Faros has an extensive set of connected data models, which allow for standardization across similar systems. In the example below you can see how a *GitLab Merge Request* (left) and a *GitHub Pull Request* (right) are treated as the same kind of entity within Faros.

GitLab Merge Request

Number	1
Author Display Name	thomas-gerber
Title	Cumque vero et et.
URL	https://gitlab.com/farosai/farosce-test/-/merge_requests/1
State	Open
Merged At	Empty
Created At	February 15, 2021, 3:54 PM
Files Changed	4
Lines Added	113
Lines Deleted	8
Repository Name	farosce-test

GitHub Pull Request

Number	311
Author Display Name	jaorr95
Title	feat: add tables for support github teams
URL	https://github.com/faros-ai/faros-community-edition/pull/311
State	Open
Merged At	Empty
Created At	August 15, 2023, 6:20 PM
Files Changed	1
Lines Added	21
Lines Deleted	0
Repository Name	faros-community-edition

Faros has standardized hundreds of objects in our GraphQL database, so you don't have to spend any effort on this internally. It is this standardization that enables us to easily populate our default libraries of dashboards and charts. The fact that we are adopted successfully at large organizations like Salesforce and Autodesk are a testament to the versatility of our solution.

For example, our data models cover the complexity and variation in deployment pipelines, such that we can stitch the data for metrics like DORA Lead Time regardless of whether:

- Some deployment pipelines know which commit SHA they are deploying
- Other deployment pipelines may only know which artifacts are being deployed, but it's the artifact repository system that knows which commit SHA was used to build the artifact

For our 70+ out-of-the-box connectors, we have pre-mapped the source data to the Faros canonical data model. As soon as you use our connectors, your dashboards will be populated.

Beyond our pre-built connectors, Faros has all the infrastructure to easily ingest homegrown systems and custom metrics *(pull)*. You can read about this in the Appendix.

Faros also has an events API *(push)* that your CI, CD or test pipelines can call easily call through a CLI, docker container or cURL command.

Attribution and Ownership

When analyzing engineering data, it is particularly important to understand which team owns which application. Faros automatically handles several types of attribution, even for mixes of monorepos and micro-services.

If you have a great service catalog available, Faros can ingest it and report when it identifies drift based on activity in your repos and applications.

If your service catalog is stale and out-of-date Faros can take the data from org charts and source control systems and create a live, up-to-date version of the catalog based on who is doing what. This helps to increase credibility and trust-worthiness of your metrics.

Jira Standardization Best Practices

We understand Jira usage within your organization might be quite heterogeneous. In fact, due to the highly customizable nature of Jira, nearly every organization we work with has wildly different workflows, statuses, fields and values.

The first way Faros can help is by putting this challenge into perspective. Engineering productivity insights are best served by constructing a picture from multiple data sources: work management, source control data, CI/CD pipeline data, incident data, HR data, cost data, etc.

Jira is indeed the tool that project and product managers go for updates and it is an important system of record. But it is not where developer experience can be found.

To get insight into the developer's flow, you need to broaden the spectrum to the critical systems where they do their work.

- Are the code changes small enough to be quickly reviewed and merged into the code base?
- Is new code being reviewed fast enough? Who is handling the reviews and why?
- Are tests passing? Which areas of the code are high risk vs. low risk?
- Is code pushed to production fast enough?
- Are we meeting our SLAs?

Faros can answer all these questions and more. These metrics do not make use of Jira data. For example, DORA metrics leverage Version Control, CI/CD and Incident Management systems.

That said, Jira is indeed the right data source to use to benchmark two key families of metrics: cycle time and planning effectiveness. **Directionally correct data is sufficient for benchmarking with minimal standardization, as we explain below.**

Rapidly Benchmarking Cycle Times and Planning Effectiveness from Jira

To rapidly benchmark against credible industry benchmarks, regardless of the development methodology (Sprint, Kanban, etc.) and specific team idiosyncrasies, there are two key metric families derived from Jira that do not require standardization: Issue Cycle Time and Planned vs. Unplanned Work.

Issue Cycle Time (and its breakdowns across various dimensions) is based on the three static status types in Jira: To Do, In Progress and Done. It is computed as the time spent between the first "In Progress" status and the first "Done" status for each Issue.

For leadership, this is a key metric.

To understand why cycle time is as long as it is, a breakdown of time spent per status — even when different custom statuses are used for different teams — will still provide directional information as to the location of current bottlenecks. For example, many will mention testing or QA, or implicate dependencies. This can be seen without standardization.

Once you understand directionally what you want to focus on, you have three options:

1. **Skip standardization altogether:** For analysis of specific areas or teams where conventions do not apply, utilize Faros's self-serve custom queries and analytics.
2. **Minimally standardize:** For a few universal metrics, adopt some simple conventions across the board in a very surgical non-disruptive manner. For example, enforce the use of a couple of new statuses.
3. **Standardize after the fact:** Standardize the data once it's in Faros using custom logic, e.g. an intermediary model. See Data Post-Processing Options for more details.

Planned vs. Unplanned Work is created based on Issue type and creation date (for issues tied to a release cadence). The mapping of Issue Types to the planned or unplanned bucket can easily be modified. The specifics of the computation are transparent and modifiable.

Concrete Example: Calculating MTTR and Incident Rate per Team

Faros's highly experienced customer success team can help your team make the right tradeoffs between accuracy (and Jira standardization) and time-to-value. Let's take an example:

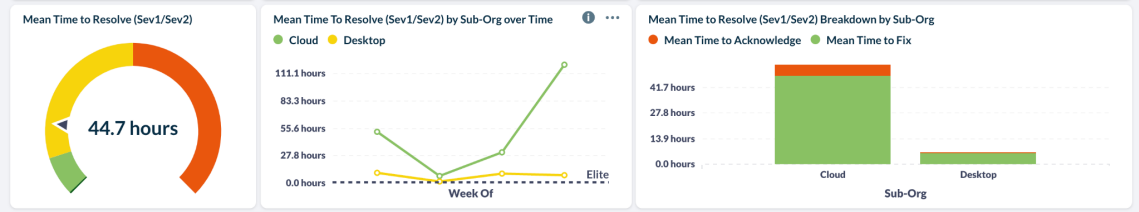
A company named Acme tracks incidents as a custom issue type in Jira ("Incident"), with a field containing the name of the failing service for attribution (assuming the existence of a service catalog).

Acme wants to compute metrics like Mean Time To Resolution (MTTR) and Incident Rate Per Team, but is concerned because the Jira 'Service' field is not consistently used and the underlying service catalog may be stale.

Faros guides Acme on a non-disruptive way to rapidly benchmark these metrics — in a way that is directionally correct, while progressively improving the data accuracy over a period of four weeks.

Here is what a typical path, guided by Faros, might look like:

1. **Week 1:** Acme uses PagerDuty, for which Faros has a prebuilt connector. With only an API token, PagerDuty data is ingested in Faros within a few hours. **Mean Time To Resolve** and **Incident Per Team** (based on the final assignee of the incident in PagerDuty) are immediately available along with an industry benchmark.



2. **Week 2:** The corresponding Jira Incidents contain the second part of the workflow, i.e., the post-mortem and fix actions. A custom flow that tracks an Incident across PagerDuty and Jira is defined in Faros, adding 'Time To Complete PostMortem' and 'Time To Fix' to the default spans already tracked ('Time to Acknowledge', 'Time To Resolve'). The attribution is still based on the reasonably trustworthy assignee field in PagerDuty, and the 'Service' custom field in Jira is not directly used.

3. **Week 3:** The Faros team advises you that **Incident Rate Per Team** is not a great metric to use because it is generally dependent on the size of the team. Therefore, Faros advises you to ingest deployment data through another standard Faros connector, which allows you to measure Change Failure Rate instead.

4. **Week 4:** Now that the metrics are available, the Acme team sees that they are pretty accurate at the sub-org or group level (VP and director, respectively), but are slightly less accurate at the team level because the assignee may not always be from the team owning the service. Faros advises Acme that the time has arrived to enforce standardization for the proper use of the custom 'Service' field in Jira moving forward, and metrics are updated in Faros to use that particular attribution model.

Effortlessly Ingesting Jira Data

Faros eliminates the need for your organization to do any of the data prep work you currently have to do to your Jira data for data lake reporting. This is truly non-differentiated data engineering that is common to any organization ingesting Jira data.

1. Faros automatically resolves Jira user identities with the rest of the organizational data. Faros can also derive team ownership of boards / projects automatically, making attribution either based on assignee or board / project trivial in dashboards.

2. Faros automatically standardizes the data in the following way for optimal analytics:
    - In the Task table, one entry per Issue per Assignee
        - It is possible to have exactly one entry per Issue with a filter
    - In the Task Board table, one entry per Issue per Board it appears on per Assignee
    - In the Task Status Detail table, one entry per Status per Issue (with the duration)
    - In the Task Release table, one entry per Issue per Release (Fix Versions)
    - In the Task Weekly Stage Count, one entry per Week per Issue Type per Stage
        - Including custom stages and custom types
    - In the Task Relationship table, one entry per Relationship between 2 Issues
    - In the Task Pull Requests table, one entry per Pull Request per Issue associated with it
    - In the Sprint History table, one entry per Issue per Sprint associated with it
    - In the Entity Additional Fields, one entry per Custom Field per Issue
3. Faros allows for the definitions of custom flows to trace processes across systems. For example, one can extend our definition of lead time to start from the moment an issue gets into an In Progress status instead of the creation of the pull request.

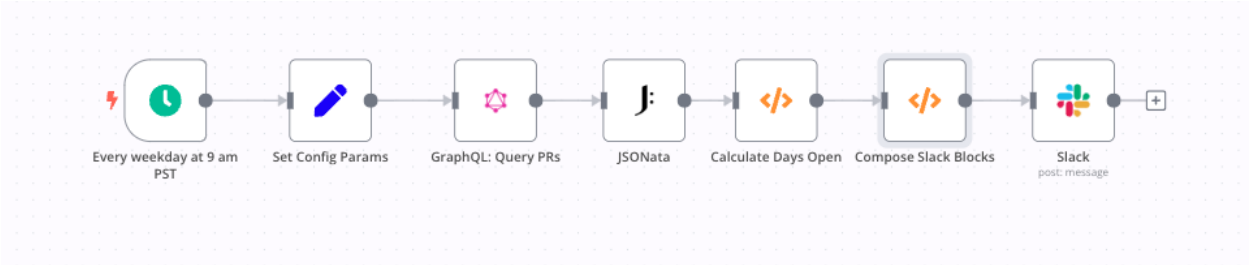
4. Idiosyncrasies and additional normalizations can be injected into reusable intermediary models.

OOTB Automation

With all your data sources connected to Faros, the next step is automation. Faros embeds n8n, a popular open-source workflow automation tool to trigger actions based on the intersection of these data sources and metrics.

You can easily create rule-based automations to:

- Send alerts and reminders
- Encourage best practices
- Create more adaptive policies



Data Post-Processing Options

Faros allows for various data post-processing options, which is useful to inject additional standardizations or idiosyncrasies.

The main options are:

- **Data layer post-processing:**
  - Transforms: Faros has a data source that can read from its graph (graphQL query), apply an arbitrary transformation (jsonata) before writing back to the graph. This option is often used to map corresponding Jira issues to Incidents when Jira acts as an Incident Management System as well.
  - [Soon] DBT: Users will soon be able to write models from the raw data using DBT.
- **Intermediary models:** Users can create "intermediary models" in the Faros BI layer, as described here.
- **Custom Flows**: A flow is a series of steps that together represent a process, which may also span several systems. Faros allows you to describe your unique flows and get metrics on them. For example: Define a Lead Time flow that includes the feature design phase. Read more here.

BI Layer with a Rich Library of Extensible Dashboards

Faros goes way beyond metrics however, and provides **full analytics** on the data it collects, which allows any area of the organization to inject team-level or organization-level idiosyncrasies into its metrics.

Faros covers a wide range of use cases and Frameworks with its pre-built libraries for:

- DevOps Management (DORA metrics)
- Engineering Productivity (including the SPACE framework)
- Software Quality
- Team Health

These libraries can be augmented and refined over time directly thanks to the Faros BI layer. It is possible to use any kind of hierarchical structure to organize the data (not just the organization chart).

For your own reporting cadences, you can create custom dashboards to reduce the lead time and effort to prep for meetings, e.g. monthly TechOps reviews, quarterly OKR check-ins, quarterly and business/product reviews.

Finally, data can be ELTed to any warehouse for further analysis.

Built-in Privacy and Security

Faros implements Role-Based Access Control. By default, metrics seen by a user are automatically scoped to the data that pertains to them or the people they manage.

Finally, Security is a core tenet of Faros: see how we remain on the leading edge of SaaS applications at security.faros.ai

Appendix: Building Custom Connectors

Faros' ingestion layer is based on Airbyte.

Writing a custom connector requires the following:

1. Writing a source that will extract the data from the desired system.
2. Write the necessary converters that will map that data onto the Faros Models.

Once that is done, those connectors can be run in an Airbyte Instance, or through a CLI.

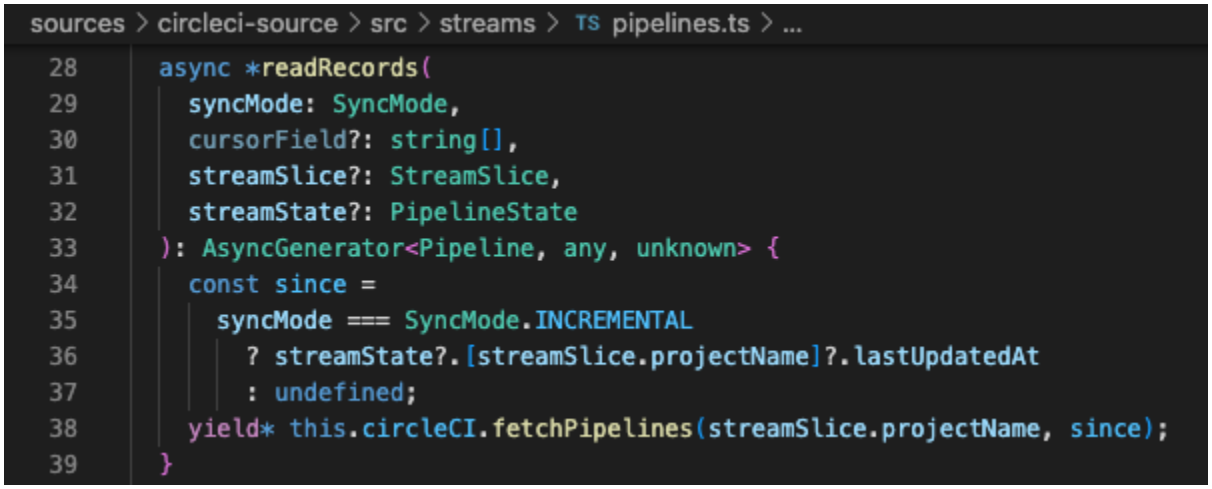
As an example, **let's take a look at our open-source CircleCI connector**.

Extracting data from the source

The recommended way to write a source is to use the Airbyte Typescript Connector Development Kit. With the CDK, it is straight-forward to make the source configurable and testable. An overview of the work is here.

In our CircleCI source, you will notice the various streams extracting corresponding data records like Projects and Pipelines.

In the screenshot below, notice how the pipeline stream emits data it extracts from the corresponding CircleCI API.



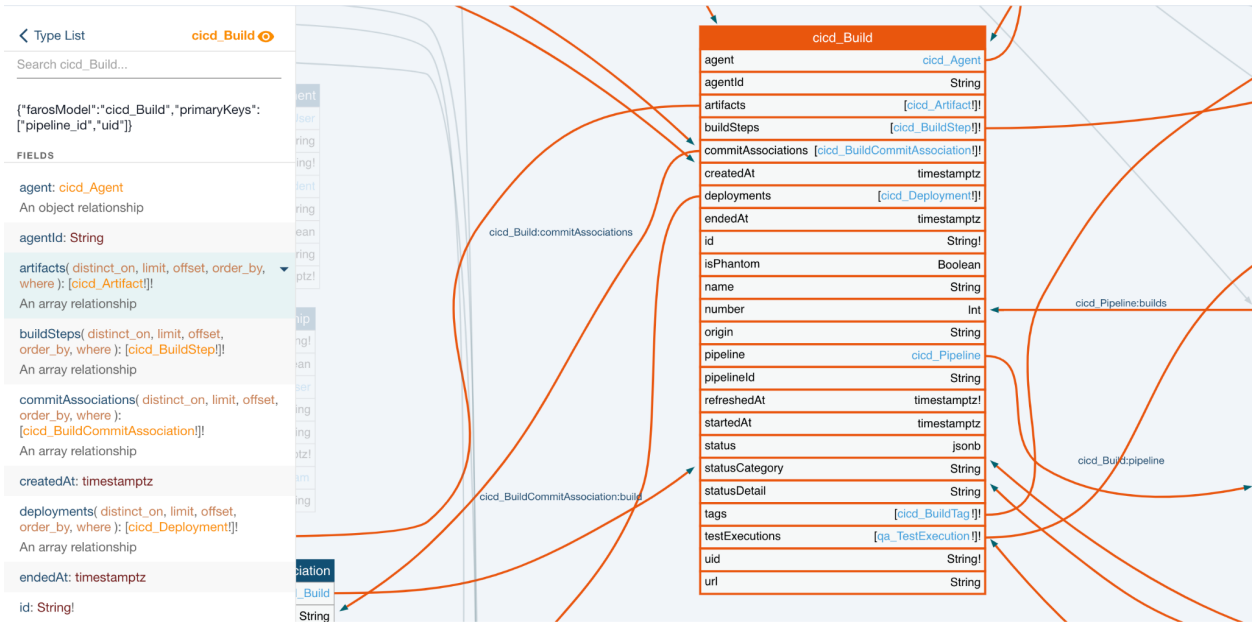
Mapping to Faros Models

The mapping is done as part of the Airbyte Faros Destination. Each stream of the source requires a converter that will convert the source record into its Faros equivalent.

In the Airbyte Faros Destination, you can find the corresponding CircleCI converters. Those map CircleCI Projects and Pipelines onto Faros cicd_Pipelines and cicd_Build models respectively.

In the screenshots below, the pipeline records extracted by the source are converted to the corresponding cicd_Build Faros entities.

```
destinations > airbyte-faros-destination > src > converters > circleci > TS pipelines.ts > Pipelines
16    async convert(
17      record: AirbyteRecord
18    ): Promise<ReadonlyArray<DestinationRecord>> {
19      const source = this.streamName.source;
20      const pipeline = record.record.data as Pipeline;
21      const res: DestinationRecord[] = [];
22
23      for (const workflow of pipeline.workflows ?? []) {
24        const buildKey = CircleCICommon.getBuildKey(
25          workflow.id,
26          pipeline.id,
27          pipeline.project_slug,
28          source
29        );
30        res.push({
31          model: 'cicd_Build',
32          record: {
33            ...buildKey,
34            number: pipeline.number,
35            name: `${CircleCICommon.getProject(pipeline.project_slug)}_${
36              pipeline.number
37            }_${workflow.name}`,
38            createdAt: Utils.toDate(workflow.created_at),
39            startedAt: Utils.toDate(workflow.created_at),
40            endedAt: Utils.toDate(workflow.stopped_at),
41            status: CircleCICommon.convertStatus(workflow.status),
42          },
43        });
```



Note that Faros Models also generically support [custom metrics] from other systems.

Running custom connectors

Those connectors can be run in an Airbyte Instance, or through a CLI. More details about options here.

For example, the CircleCI connector can be run through this command.

CircleCI

Shell

```
./airbyte-local.sh \
  --src farosai/airbyte-circleci-source:latest \
  --src.token $CIRCLECI_API_TOKEN \
  --src.project_names '["vcs-slug/org-name/repo-name-1", ...]' \
  --src.cutoff_days 90 \
  --dst farosai/airbyte-faros-destination:latest \
  --dst.edition_configs.api_key $FAROS_API_KEY \
  --dst.edition_configs.api_url "https://prod.api.faros.ai" \
  --dst.edition_configs.graphql_api "v2" \
  --dst.edition_configs.edition "cloud" \
  --connection-name "mycirclecisrc"
```