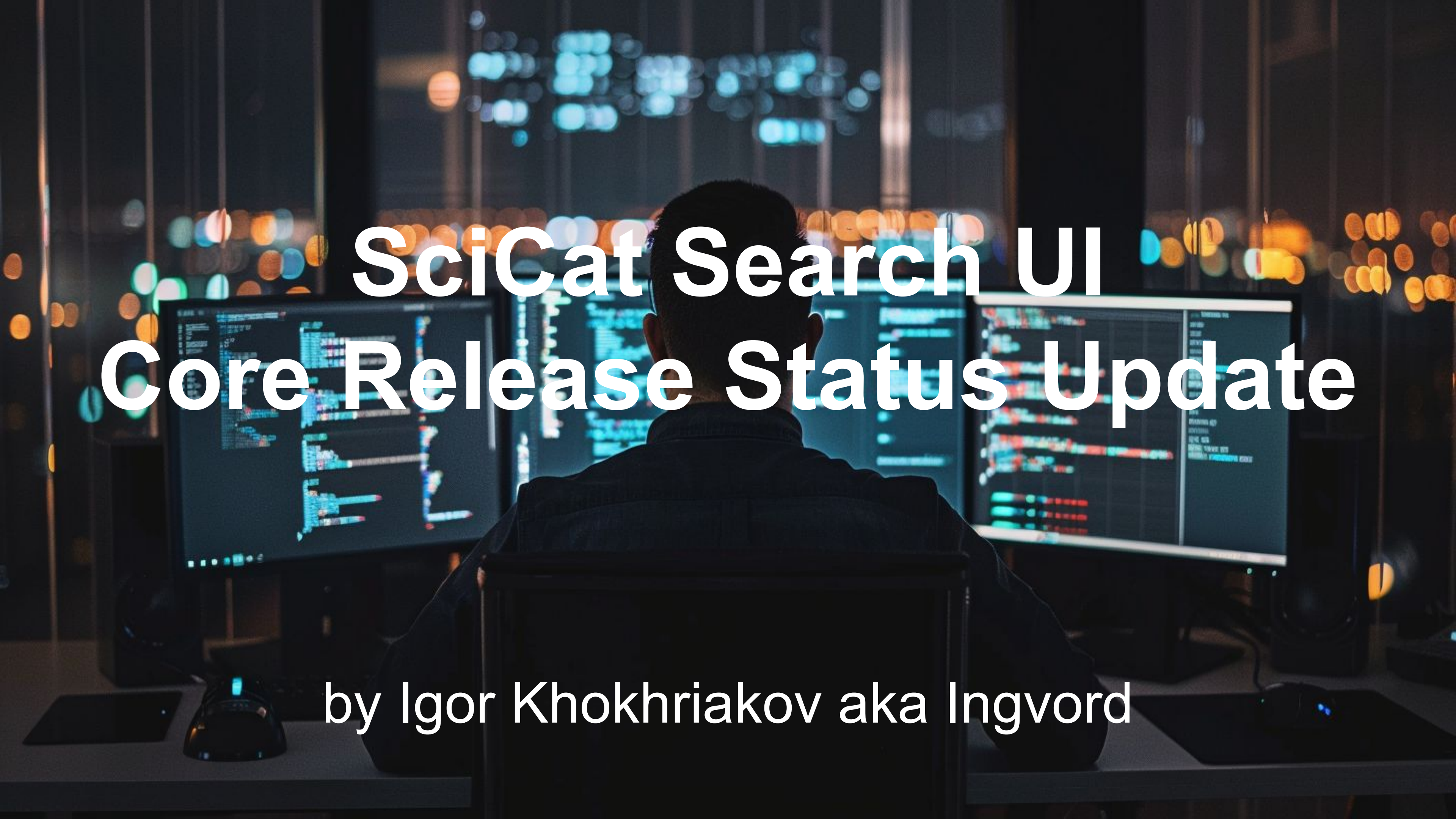
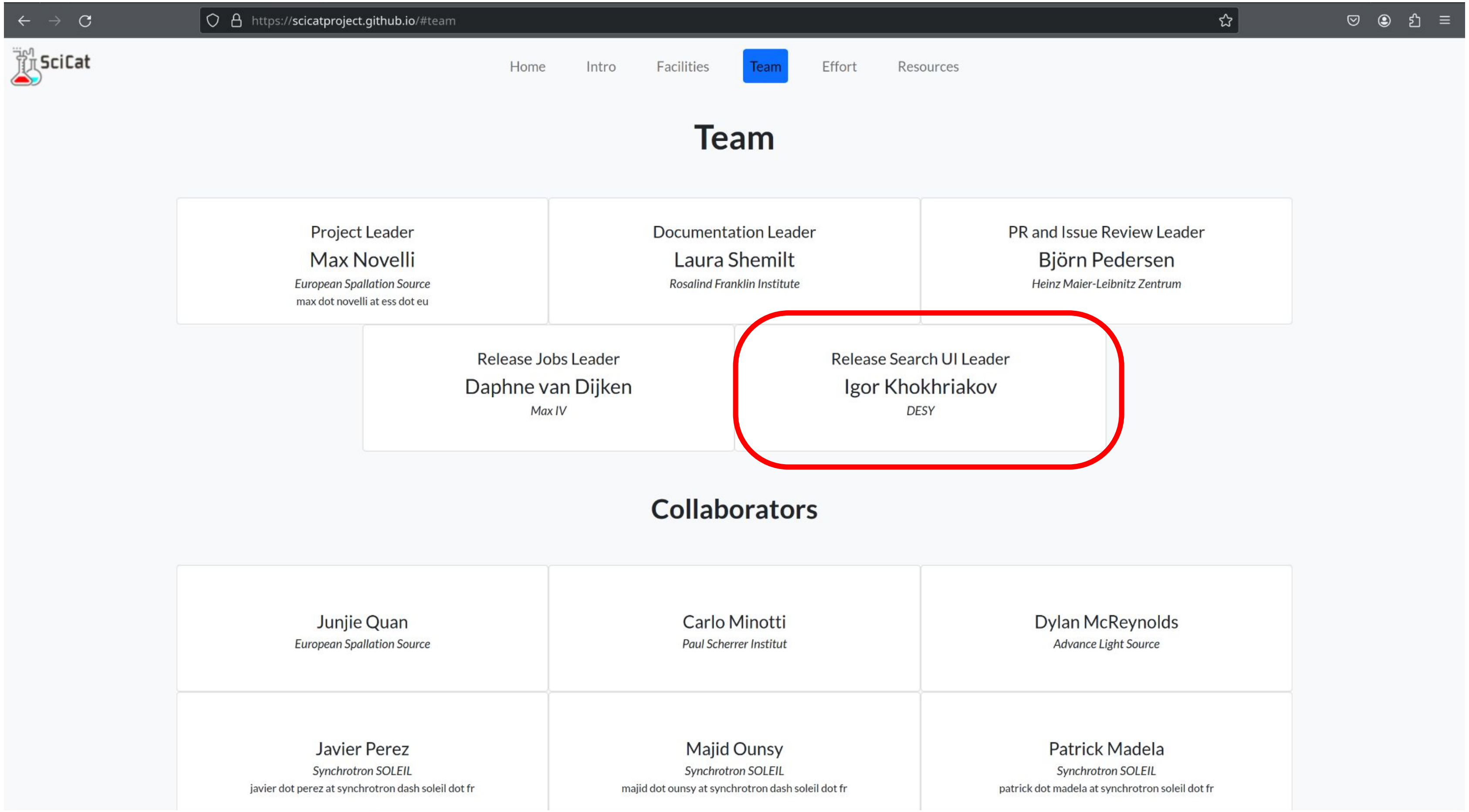


SciCat Search UI Core Release Status Update

by Igor Khokhriakov aka Ingvord

Embrace serverless design
Easy to share (Helm)
Opens window to integrate 3rd party open s
tools e.g. observability, service meshes etc
I ARE PROGRAMMER
I MAKE COMPUT
BEEP BOOP BEEP BEEP B
Ubuntu Summit 2023

- Scientific Software Architect at Hereon/UCSD/DESY
- Tango-Controls Kernel Developer
- ~800 citations on Google Scholar
- >3K followers on LinkedIn



https://scicatproject.github.io/#team

SciCat

Home Intro Facilities Team Effort Resources

Team

Project Leader
Max Novelli
European Spallation Source
max dot novelli at ess dot eu

Documentation Leader
Laura Shemilt
Rosalind Franklin Institute

PR and Issue Review Leader
Björn Pedersen
Heinz Maier-Leibnitz Zentrum

Release Jobs Leader
Daphne van Dijken
Max IV

Release Search UI Leader
Igor Khokhriakov
DESY

Collaborators

Junjie Quan
European Spallation Source

Carlo Minotti
Paul Scherrer Institut

Dylan McReynolds
Advance Light Source

Javier Perez
Synchrotron SOLEIL
javier dot perez at synchrotron dash soleil dot fr

Majid Ounsy
Synchrotron SOLEIL
majid dot ounsy at synchrotron dash soleil dot fr

Patrick Madela
Synchrotron SOLEIL
patrick dot madela at synchrotron soleil dot fr

Search UI Release Focus:

- **Enhanced User Experience**: Develop an intuitive and configurable interface for efficient dataset management and navigation.
- **Personalized Settings**: Implement a robust backend to support user-specific settings for facets, columns, and search preferences.
- **Interactive Search Interface**: Provide a prominent full-text search bar, dynamic filter tags, and an interactive display of search results, requiring user action to load data.
- **User-Centric Design**: Ensure the interface serves to individual user needs, with the ability to save configurations and reset to defaults, enhancing overall usability and satisfaction.

Interactive Search Interface

- Design a full-text search feature that spans the entire viewport, emphasizing its importance and accessibility (#1148).
- Implement filter tags that display current search criteria, allowing users to understand and modify their search context easily (#1133&#1148).
- Require user action to initiate the search, avoiding automatic loading of potentially irrelevant datasets (#1149).
- Integrate an 'Apply' button to trigger the search process, reinforcing user control over data retrieval (#1149).

Personalized Settings (#604)

- Expand the `userSettings` endpoint to include user-specific configurations for columns, filters, and metadata sections.
- Introduce an admin-accessible endpoint for default settings, providing a foundation for anonymous users and initial user setups.
- Enable users to store their preferences for visible columns and filter facets within the main Scicat database.
- Ensure that personal settings are dynamically applied across sessions, offering a consistent and tailored user experience.
- **Develop a user-friendly interface for settings adjustment, including a reset option to revert to default configurations.**

User-Centric Design

- Ensure that UI elements like the settings icon are conveniently placed for quick access to configuration options (#1133&#1132).
- **Design modal dialogs that can be invoked from various UI locations, adapting to different user workflows (#1132&#1141).**
- Implement a blank initial state for dataset lists, prompting users to actively search or filter to view data (#1149).
- Present a clear and informative message when no datasets are displayed, guiding users towards using search or filter functions (#1149).
- Optimize the metadata filter widget to accommodate various data types, providing appropriate matching options for an enhanced search experience (#1141).

Summary: 40% progress of the project has been achieved

Live demo…

Enhanced User Experience (#614)

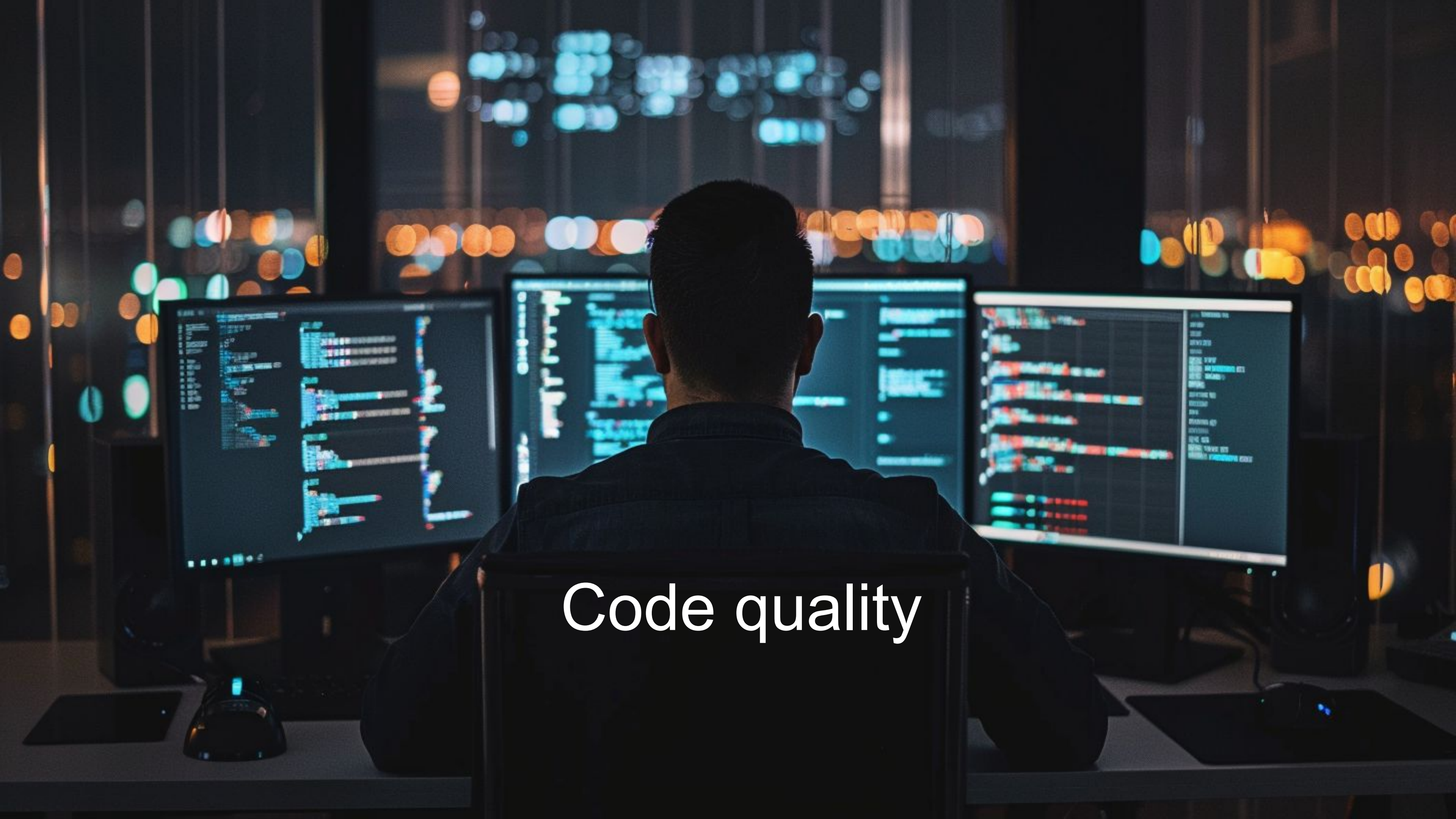
- Pre-populate the configuration view with metadata from accessible datasets, offering a smart starting point for user interaction.
- Guess data types for scientific metadata to streamline user configuration processes.
- Merge high-level field configurations with metadata keys for a unified configuration approach.
- Facilitate decisions on whether front-end or back-end should handle configuration merging, enhancing system efficiency.
- Determine admin capabilities for overwriting default configurations, ensuring flexibility and control in data presentation.

Established a collaboration

Sci Cat

1850 MELIORA

UNIVERSITY *of* ROCHESTER



Code quality

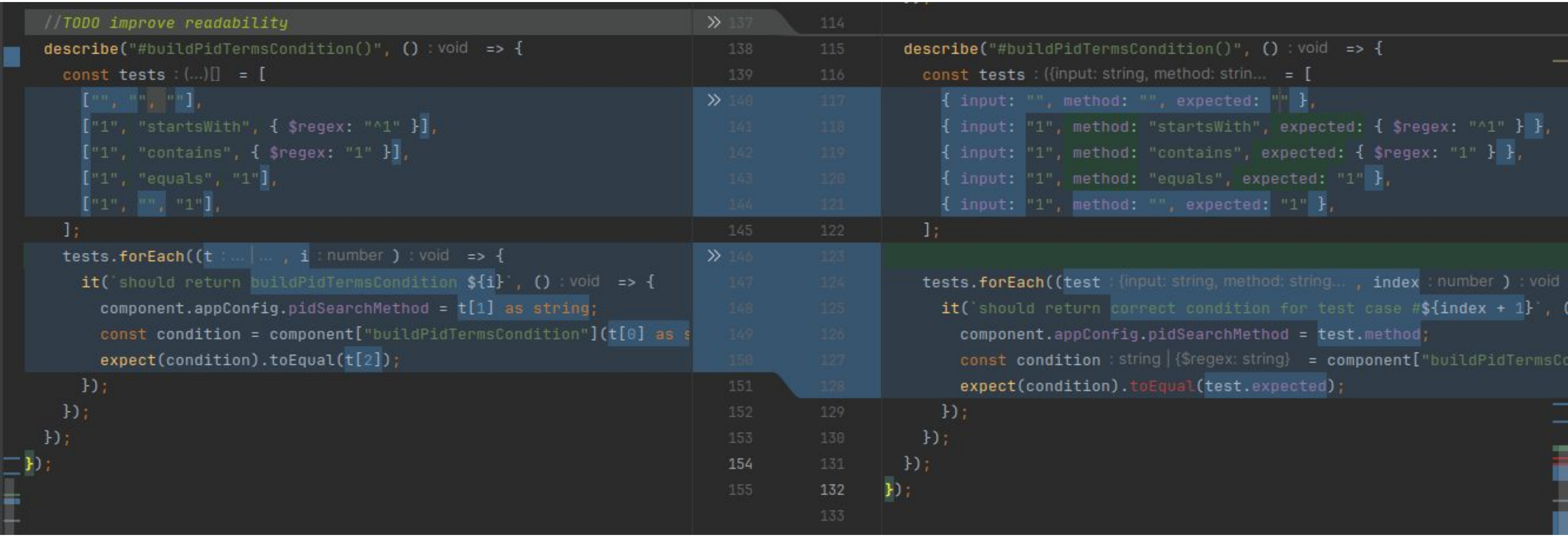
Single responsibility

See e.g. Dataset-filter.component.ts, PR #1465 or Pagination in #1503

Live Demo…

Readability

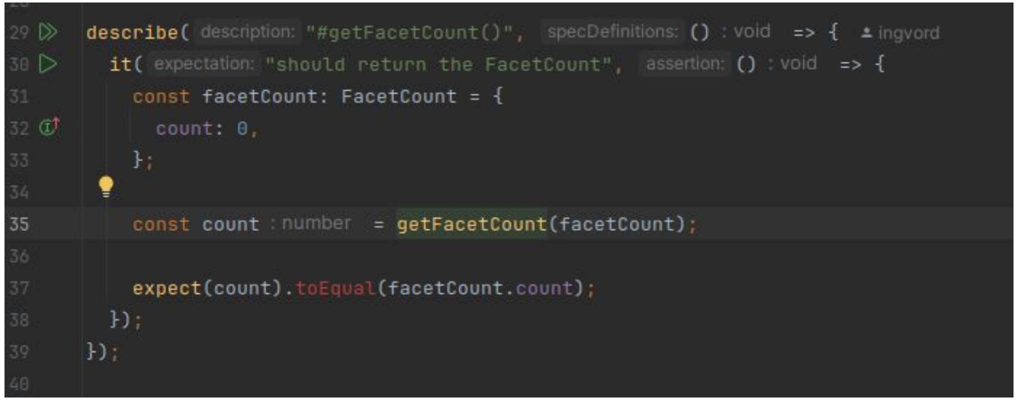
See e.g. pid-filter.component.ts in Pr #1465



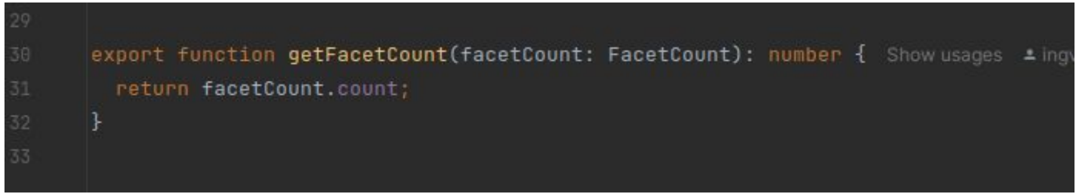
```
//TODO improve readability
describe("#buildPidTermsCondition()", () : void  => {
  const tests : (...)[]  = [
    ["", "", ""],
    ["1", "startsWith", { $regex: "^1" }],
    ["1", "contains", { $regex: "1" }],
    ["1", "equals", "1"],
    ["1", "", "1"],
  ];
  tests.forEach((t : ... | ... , i : number ) : void  => {
    it(`should return buildPidTermsCondition ${i}`, () : void  => {
      component.appConfig.pidSearchMethod = t[1] as string;
      const condition = component["buildPidTermsCondition"](t[0] as s
      expect(condition).toEqual(t[2]);
    });
  });
});
});
```

```
describe("#buildPidTermsCondition()", () : void  => {
  const tests : ({input: string, method: strin...  = [
    { input: "", method: "", expected: "" },
    { input: "1", method: "startsWith", expected: { $regex: "^1" } },
    { input: "1", method: "contains", expected: { $regex: "1" } },
    { input: "1", method: "equals", expected: "1" },
    { input: "1", method: "", expected: "1" },
  ];

  tests.forEach((test : {input: string, method: string... , index : number ) : void
    it(`should return correct condition for test case #${index + 1}`, (
      component.appConfig.pidSearchMethod = test.method;
      const condition : string | {$regex: string}  = component["buildPidTermsCo
      expect(condition).toEqual(test.expected);
    });
  });
});
});
```

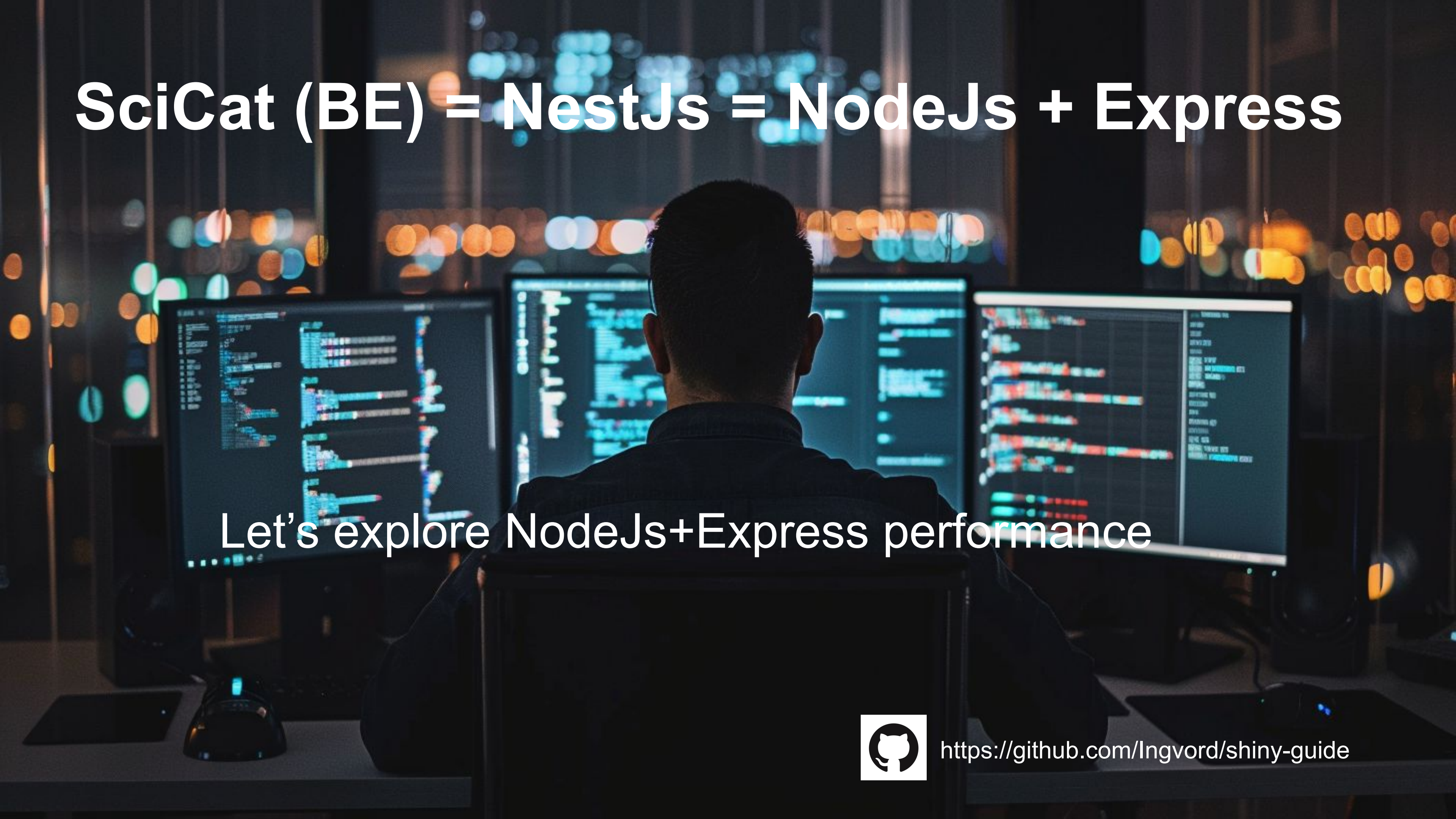
Avoid Trivial Tests: facetsCount



```
describe( description: "#getFacetCount()",  specDefinitions: () : void  => {  ingvord
  it( expectation: "should return the FacetCount",  assertion: () : void  => {
    const facetCount: FacetCount = {
      count: 0,
    };

    const count : number  = getFacetCount(facetCount);

    expect(count).toEqual(facetCount.count);
  });
});
```



```

export function getFacetCount(facetCount: FacetCount): number {  Show usages   ingv
  return facetCount.count;
}

```



SciCat (BE) = NestJs = NodeJs + Express

Let’s explore NodeJs+Express performance

https://github.com/Ingvord/shiny-guide

Prerequisites

All tests were performed on a typical single-instance virtual machine, armed with **8 CPU** cores and **12 GB RAM**

Wrk2 was used to simulate requests*

```
wrk -R{1000..10000} -t10 -c1000 -d30
```

-R – rate

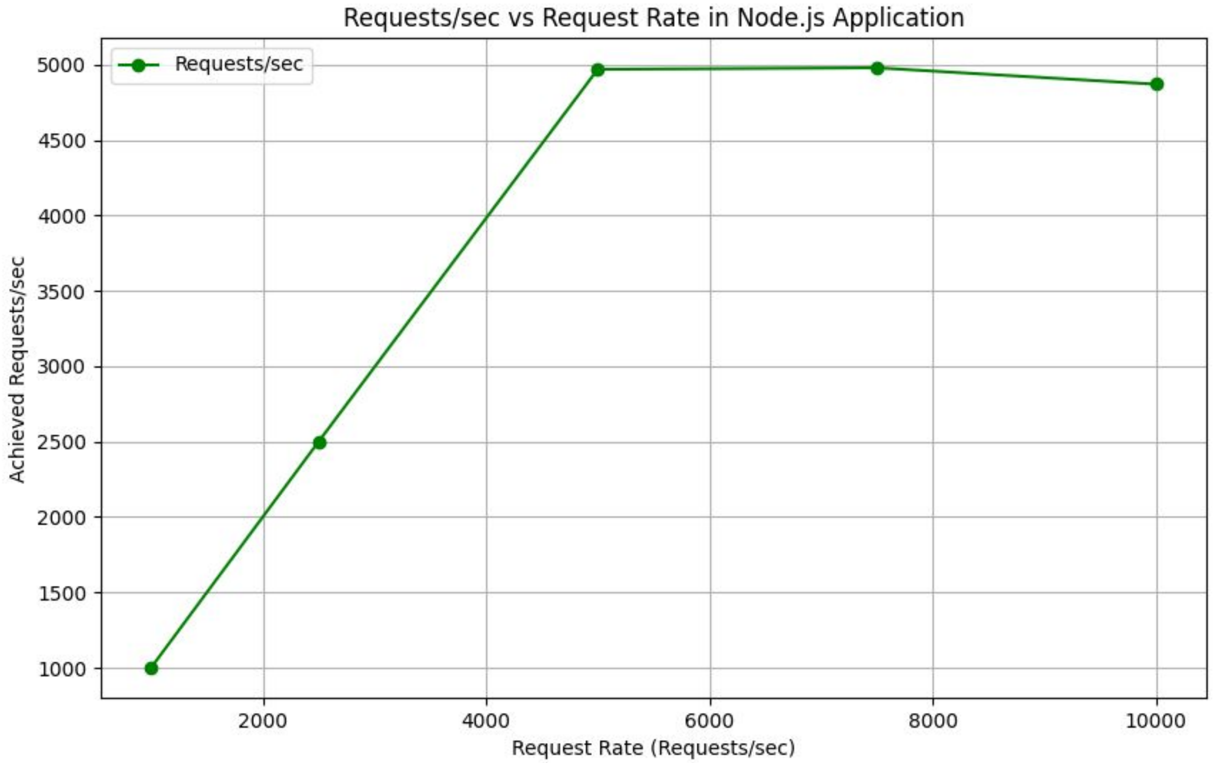
-t – number of threads

-c – connections

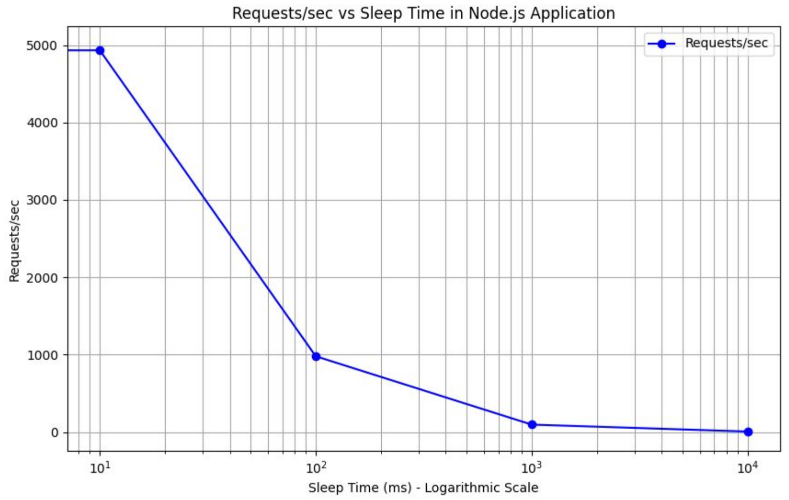
-d – duration

Above simulates how 10_000 clients requesting during 30s with various rate

Baseline: 5_000 RPS

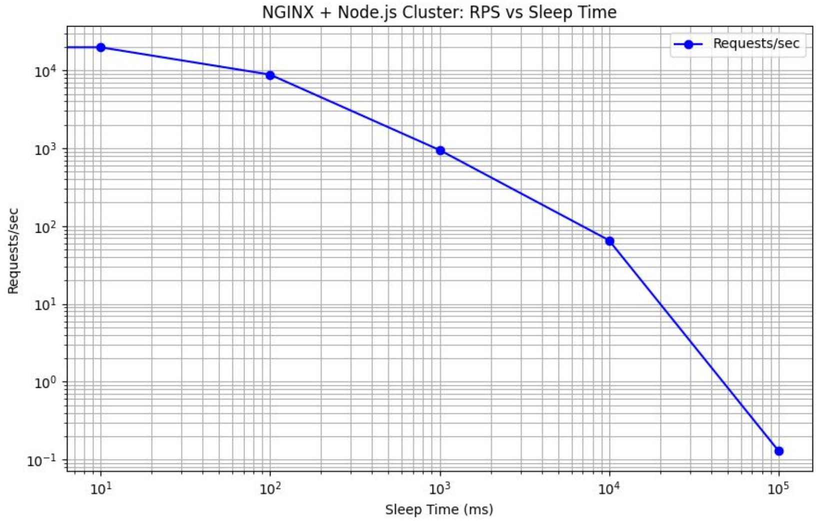


IO load (single instance*, 8 instances + nginx)



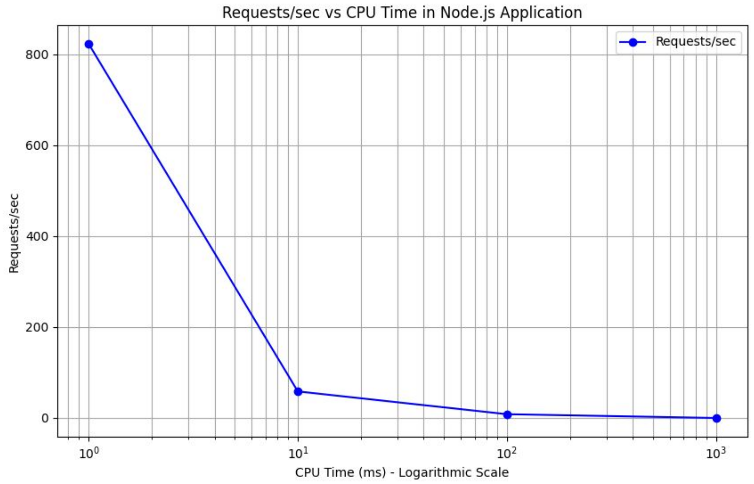
Single instance

8 instances + nginx



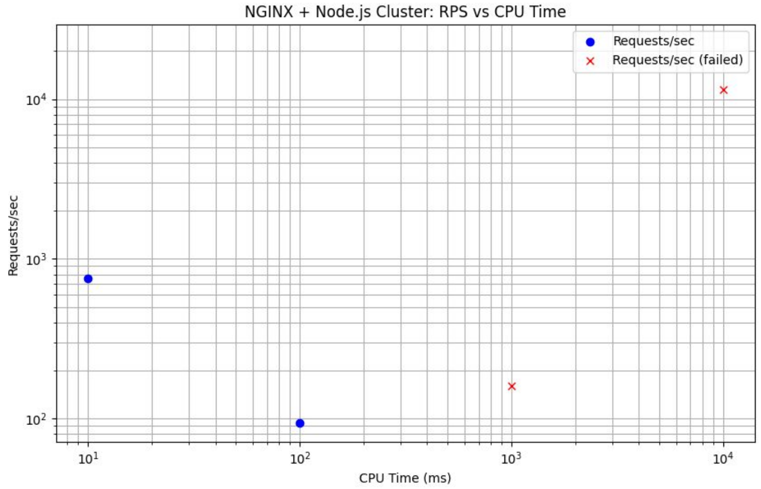
* instance here and below means a process running on the VM

CPU load (single instance, 8 instances + nginx)



Single instance

8 instances + nginx



BONUS:

Webix widgets integration

Thanks!

Questions?

Igor.Khokhriakov@desy.de

https://www.linkedin.com/in/ikhokhryakov/
https://ingvord.ru