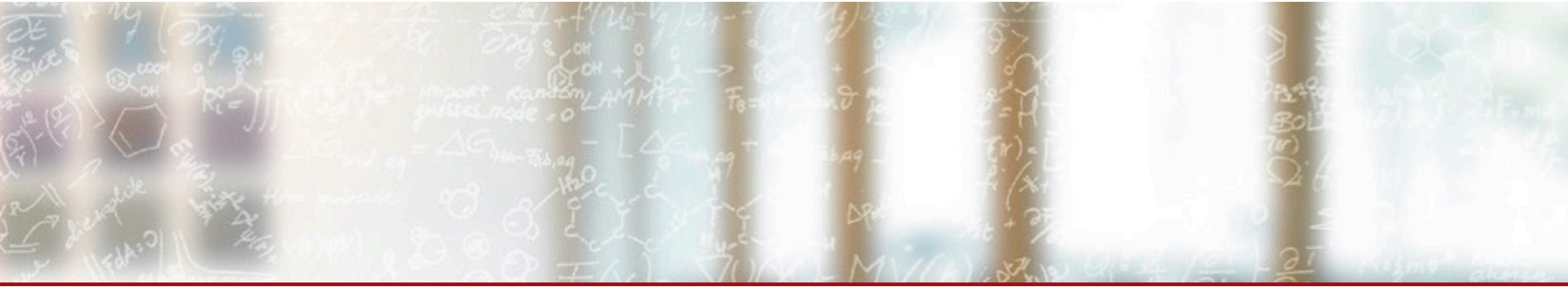
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Divide and Rule
Automated Workload Distribution for Efficient User Support Services

CUG2025 Technical Session 1B, May 6 2025

Luca Marsella, ETH Zürich / Swiss National Supercomputing Centre (CSCS)

Outline

- User Services
  - **Service Catalogue** and **Support Policies**

- **Working structures**
  - Implementing Site Reliability Engineering

- How users get support
  - **Knowledge Base** and **Service Desk**

- Community **Slack** Space and **Status Page**

Motivation

Organizational transformation

- Improve **performance** of operations and **service delivery**
- Cultural shift toward **Agile and DevOps** methodologies
- Functional units with **end-to-end responsibility** on a set of services

Change of **roles** and **tools**

- **Distribute** user support across functional units
- Replace weekly roster by increasing **automation**
- Adopt **project management tools** to organize daily tasks

User Services: Service Catalogue and Support Policies

Service Catalogue

List of services developed and maintained by CSCS Working Structures

- **Infrastructure Services**
  Computing and Storage Resources
- **Platform Services**
  Authentication, Containers, CI/CD, Workload Manager,…
- **User Environment**
  Programming Environment with CPE and CSCS **uenv**

- Each service is maintained by a specific Working Structure
  Yellow pages with WS Service/Tools/Micro-service responsibility
  Specific list of responsibilities for the Slurm Workload Manager

Support Policies

CSCS offers timely response with **direct assistance** or **escalation processes**

- Support is **limited to the scope** of project proposals
- CSCS cannot guarantee the resolution of all issues

Supported community codes and user applications

- CSCS helps users run up-to-date supported community codes
  CSCS staff cannot fix application-specific issues
- Best-effort support deploying and optimizing user applications

Prioritization case impact, complexity, knowledge transfer, time to solution

Collaborative support consult online docs and provide detailed information

Working Structures

Working Structures

CSCS organizes its work into **Working Structures (WS)**

- Each WS is a functional unit with defined purposes and outcomes
- Functional units developing and maintaining production services

CSCS staff are members of at least one and maximum two WS

- They have specific roles and responsibilities within each WS
- Each WS has team member with the role of WS **leader**

Working Structures are a type **Workgroup**, **Panel** or **Project**

- Big WS can have Sub-Workgroup type to facilitate work
- Sub-working structures cannot be divided further

Site Reliability Engineering

Site reliability engineering (SRE) is a software engineering approach to IT

- Use software as a tool to **monitor** systems and identify issues
- **Automate** operative tasks to manage **scalable** and **reliable** systems

CSCS journey to SRE started recently

- Definition of **service-level objectives** (SLO) for system availability
- Monitoring **service-level indicators** (SLI) to measure system uptime

Working Structure **Service Reliability Management** (SRM)

- Manage CSCS services, ensuring that they meet their goals
- Introduce SRE practices in a Service Oriented Architecture (SOA)
- **Measure** and **monitor** services **automatically** to react promptly

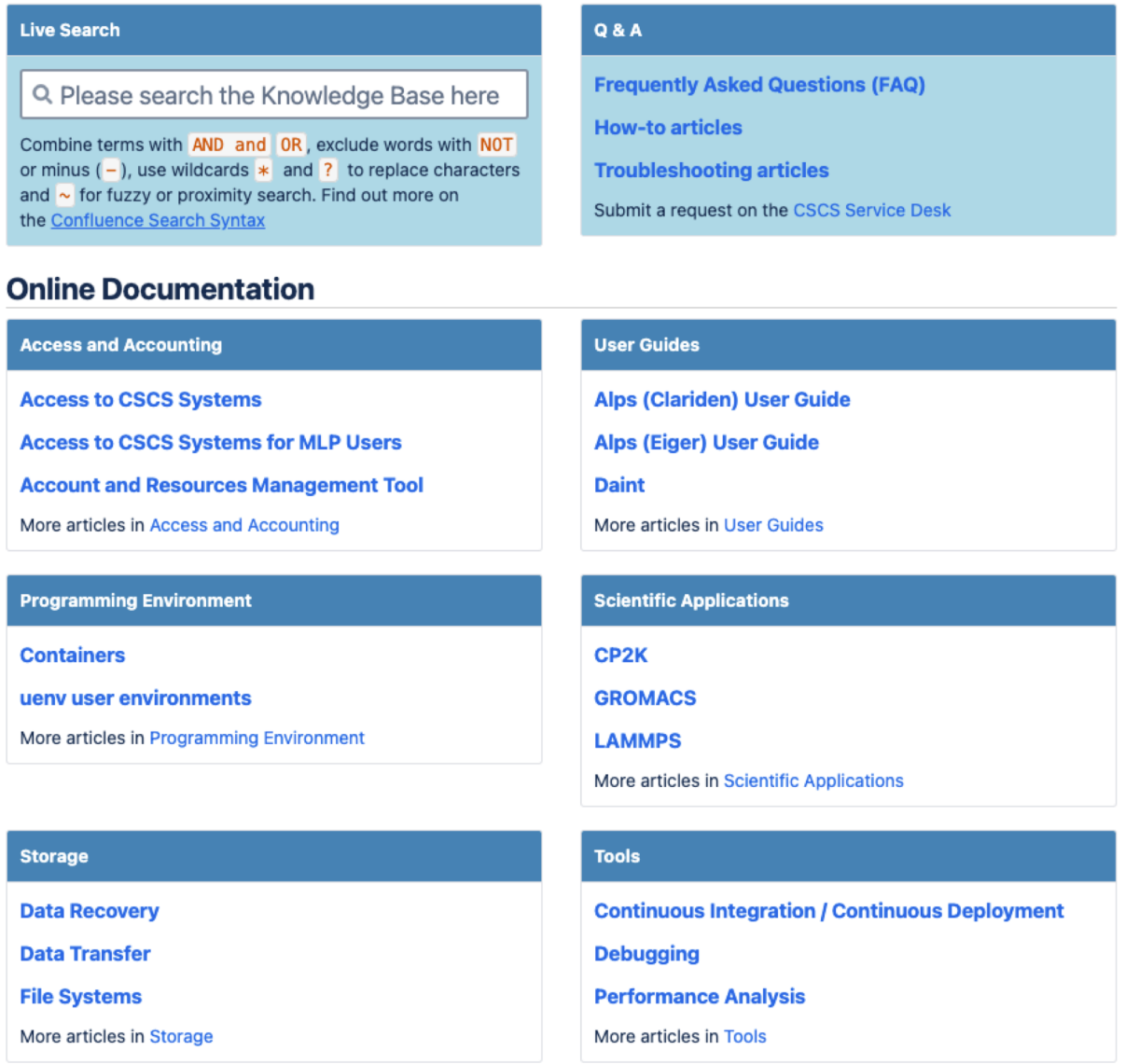
Tools

- Atlassian Jira and Confluence (Data Center, self-managed)
  Jira Service Management (JSM) and Confluence Knowledge Base (KB)
- elasticsearch — Grafana and Kibana dashboards
- GitHub GitLab — Pull and merge requests
- OneUptime — Status page
- slack — Notifications from Atlassian tools integrations
- sympa — Mailing List Management Software

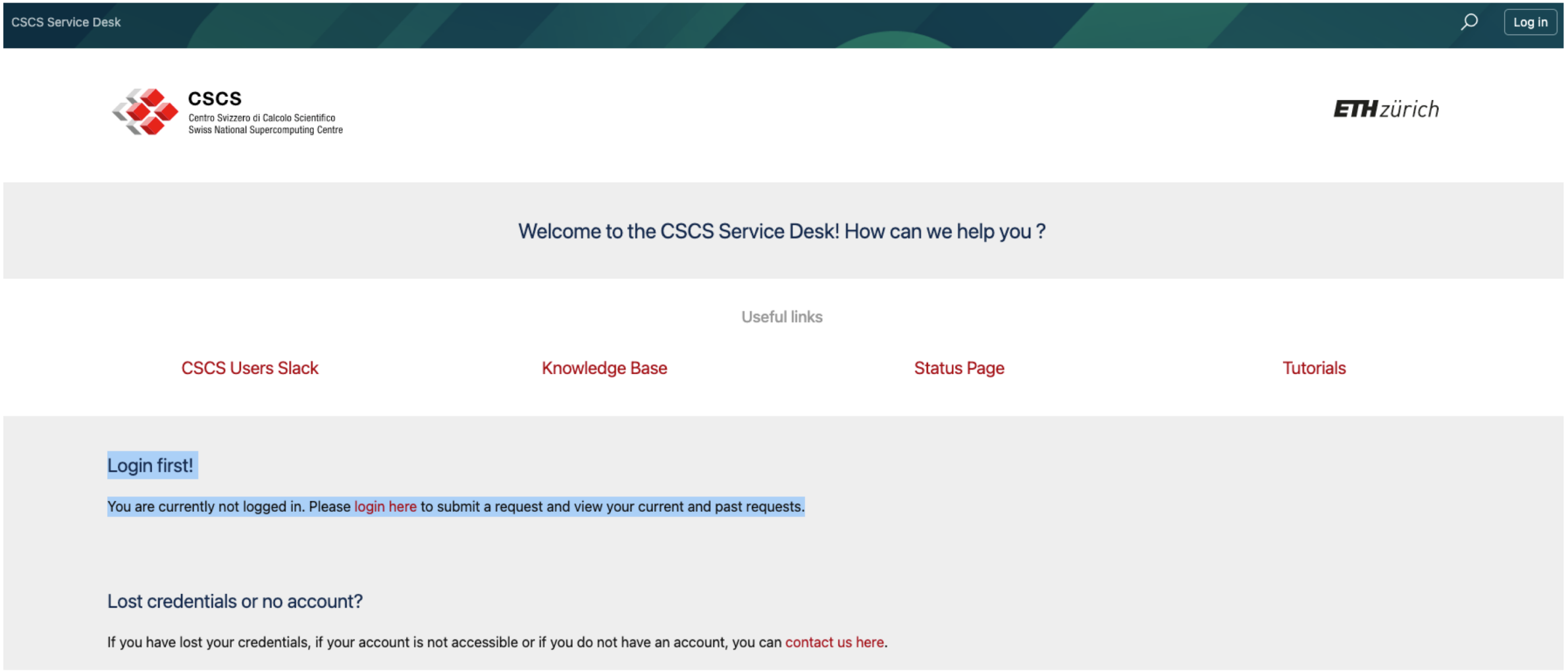
Knowledge Base and Service Desk

CSCS Knowledge Base (KB)

- Currently hosted by Confluence KB, **content migration to MkDocs**
- **Matching articles** are displayed in the search bar of the JSM
- The **training program** featuring tutorials recordings should also be indexed for search in the KB



CSCS Service Desk powered by Jira Service Management (JSM)



How users submit a support request

Users can submit requests to the CSCS Service Desk in two ways

- On the JSM web interface upon authentication
- Using the online form **contact us** without log in

The first option is the preferred one and users are encouraged to use it

- User requests can be **tracked** and the **user project** is recorded
- Requests can be filtered with **request types** mapping Working Structures

The second option does not gather the same information on users

- Users **identity and project** need to be set manually if email don't match

A Knowledge Base article explains **How to submit a support request**

Mandatory data of a CSCS support request

Summary

- Matching articles that might help will pop-up from the KB as you type

Description

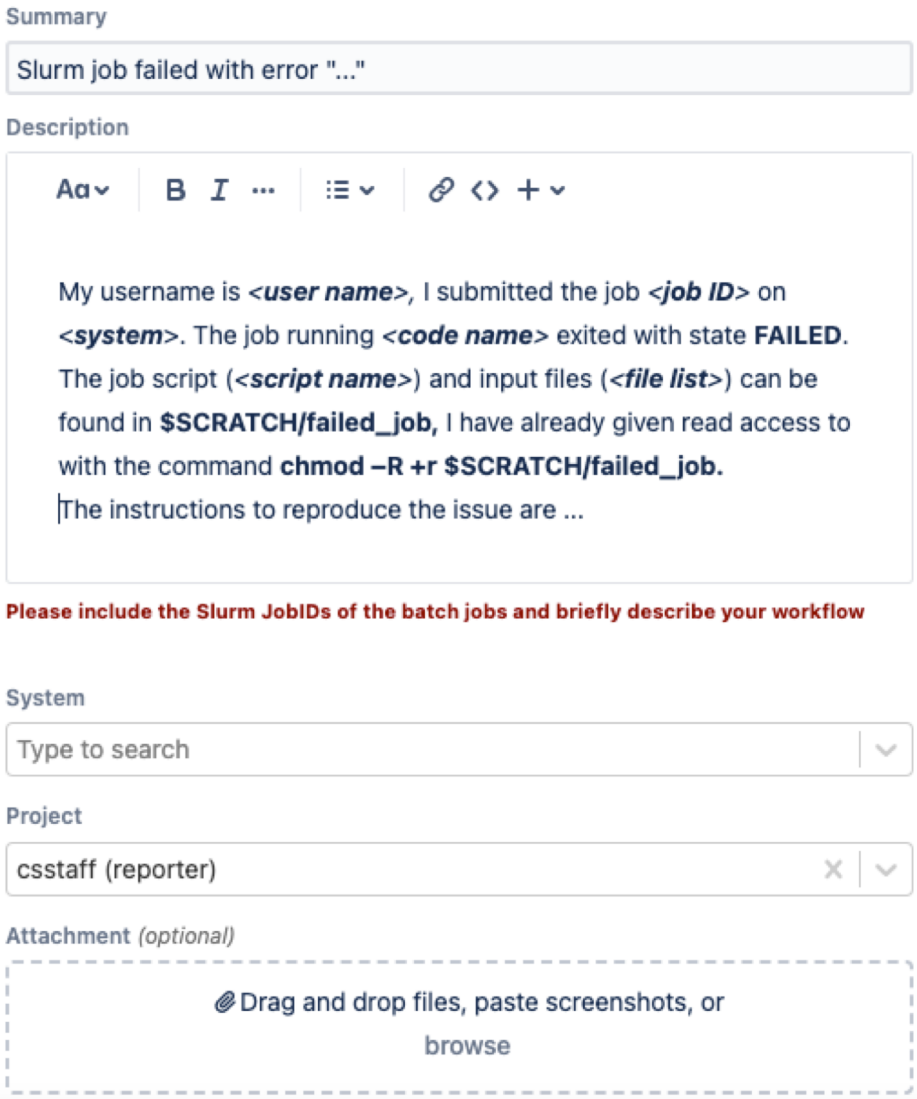
- Users provide the information needed to **reproduce the issue**

System

- The cluster used in production

Project

- Account providing compute budget



Classification of support requests

Text classification to categorize user requests

- **Automation rules** route requests to the appropriate Working Structure

Compared **rule-based** versus **machine learning-based** text classification

- Selected top **unique keywords** by frequency in the request summary
- Standard NLP model from Hugging Face (microsoft/deberta-v3-large)

Conclusion

- Correct classification: up to ~ 51 % rule-based vs ~ 70% ML-based
- Rule-based is easy to implement with **JSM automation**
- ML-based requires **external interface** with JSM API (without plugins)

Thanks to my colleague **Henrique Mendonça** for performing the comparison

Caveats of rule-based automation

Expertise in categories is required to create effective rules

- Keywords need **frequent updates** with **human intervention**
- Rules **do not scale efficiently** with overlapping content
- Relatively **low sensitivity**, contextual search improves performance

Additional details are often required to classify correctly user requests

- **Project** linked to the account providing compute budget
- **System** defining the vCluster used in production

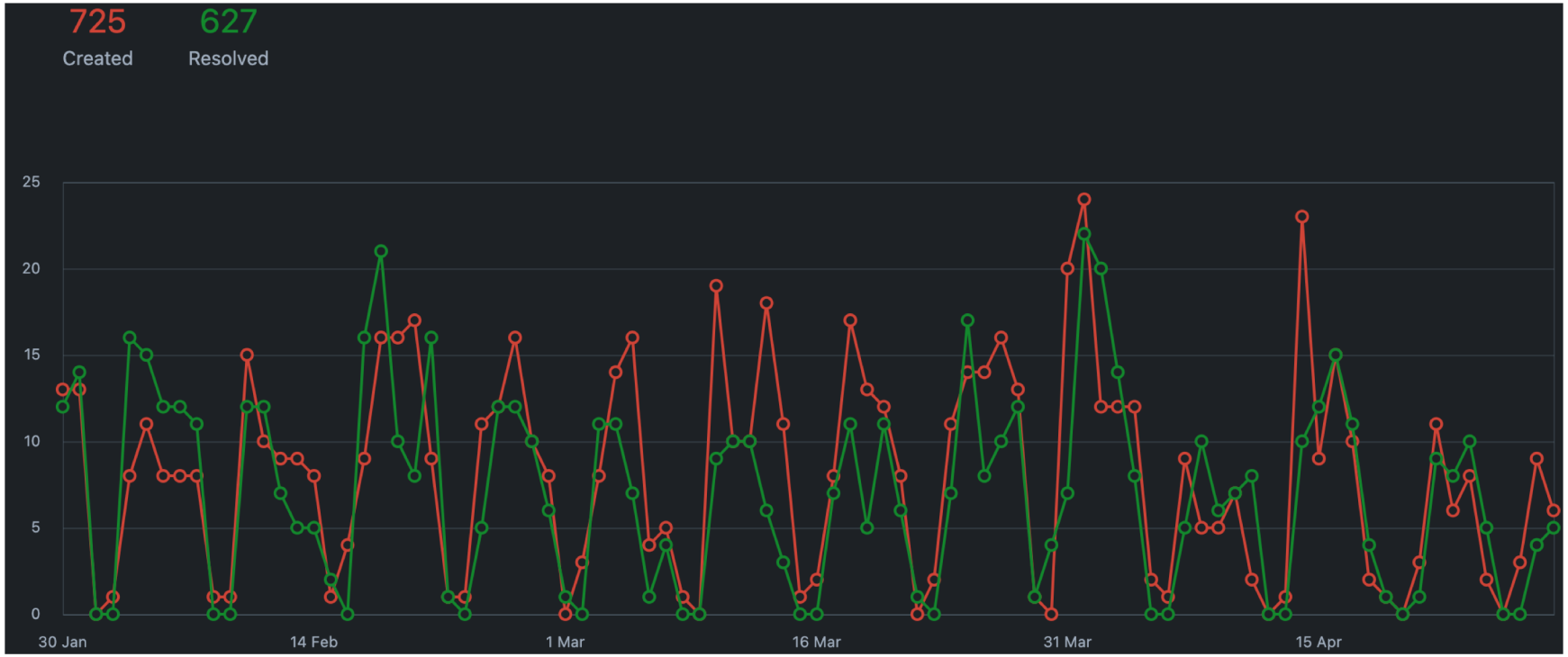
The information comes in the corresponding mandatory fields

- Additional automation rules **integrate** the keyword search in the summary

Impact of user requests

Inspecting JSM reports

- Relatively **limited number** of requests
- **Direct mapping** of requests to WS feasible with few **keywords** search
- **Reduced** need of **rule-based** request **classification**

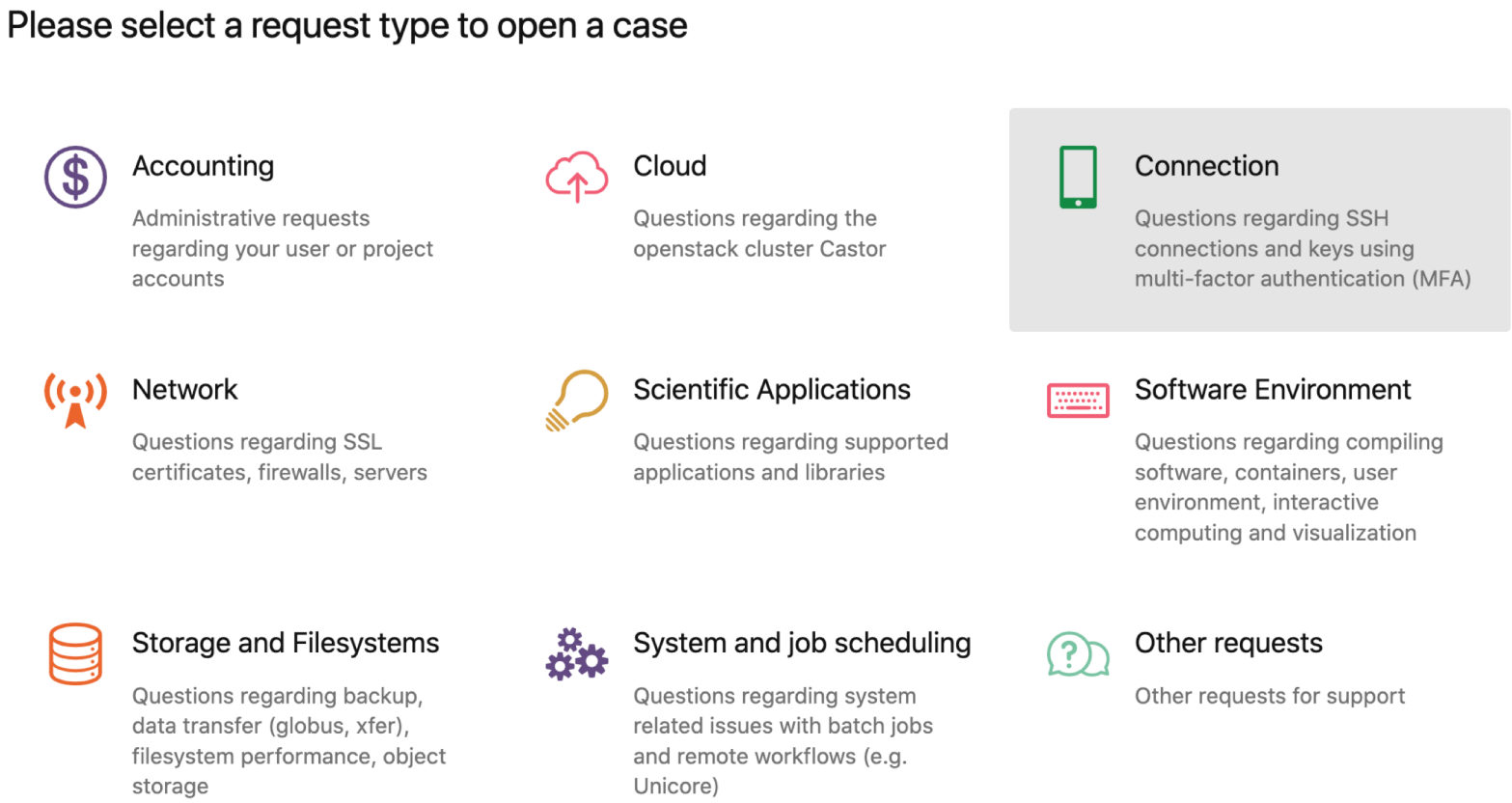


CSCS Service Desk user requests (created vs. resolved) in the past quarter by day

Request types in the CSCS Service Desk

Requests types are only available after users log in

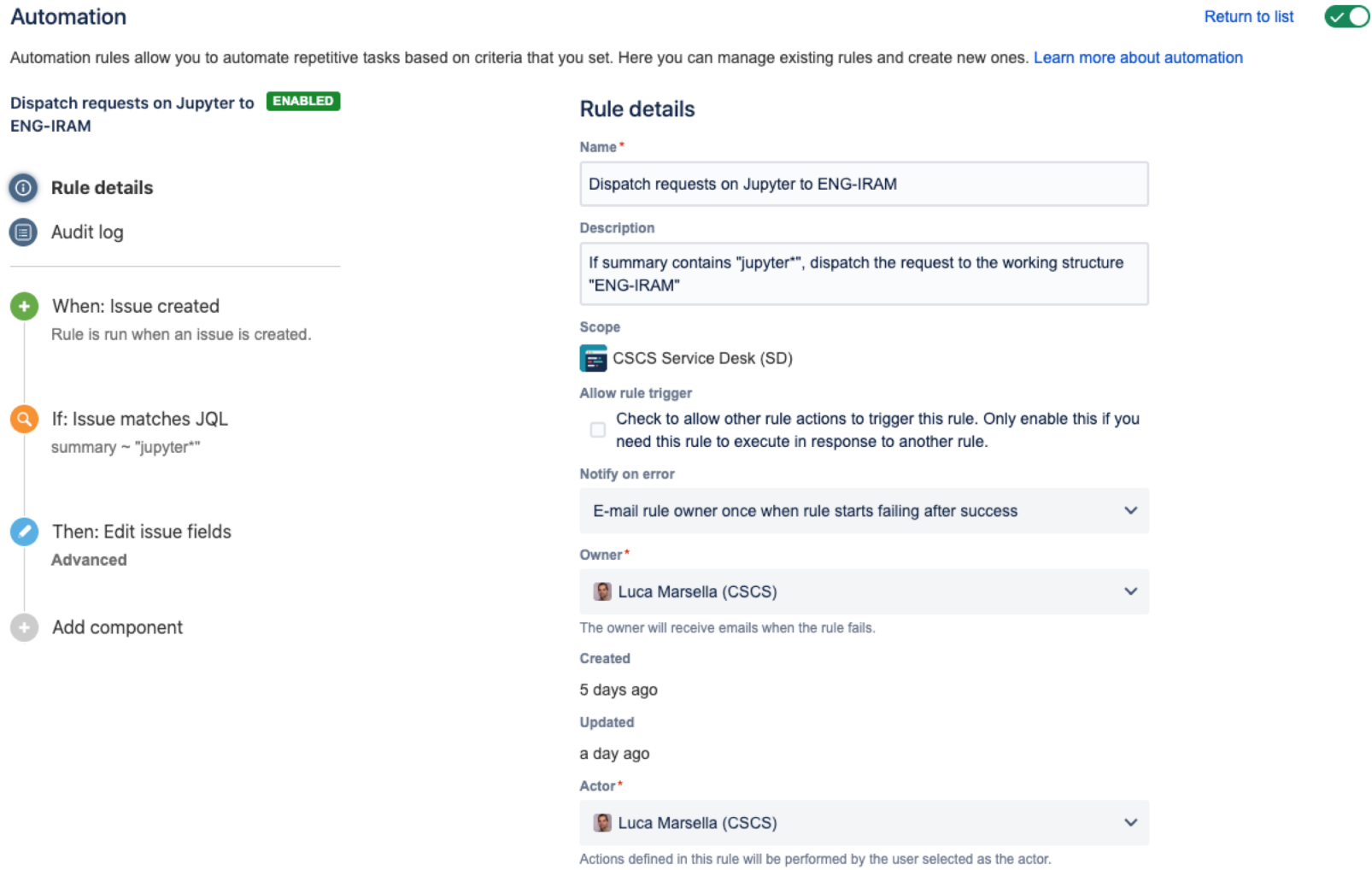
- Few keywords define each **request type**
- Requests can be submitted if **mandatory fields** are filled in
- The request are directly mapped to a single WS



Example of JSM automation rule

Scope of the automation rule

- Dispatch inquiries on **Jupyter** to ENG-IRAM
- The rule is executed as soon a **new request** is created in JSM
- Search the keyword **jupyter*** in summary
- The request is routed to ENG-IRAM if a **match** is found



Request assignment

Known issues with direct mapping based on request types

- Users might not choose the request type appropriately
- No direct mapping available for **Other requests** and **E-mails**

WS assess and review the requests assigned

- Select **assignee** among WS members if appropriate
- **Re-route** the request to the WS in charge otherwise

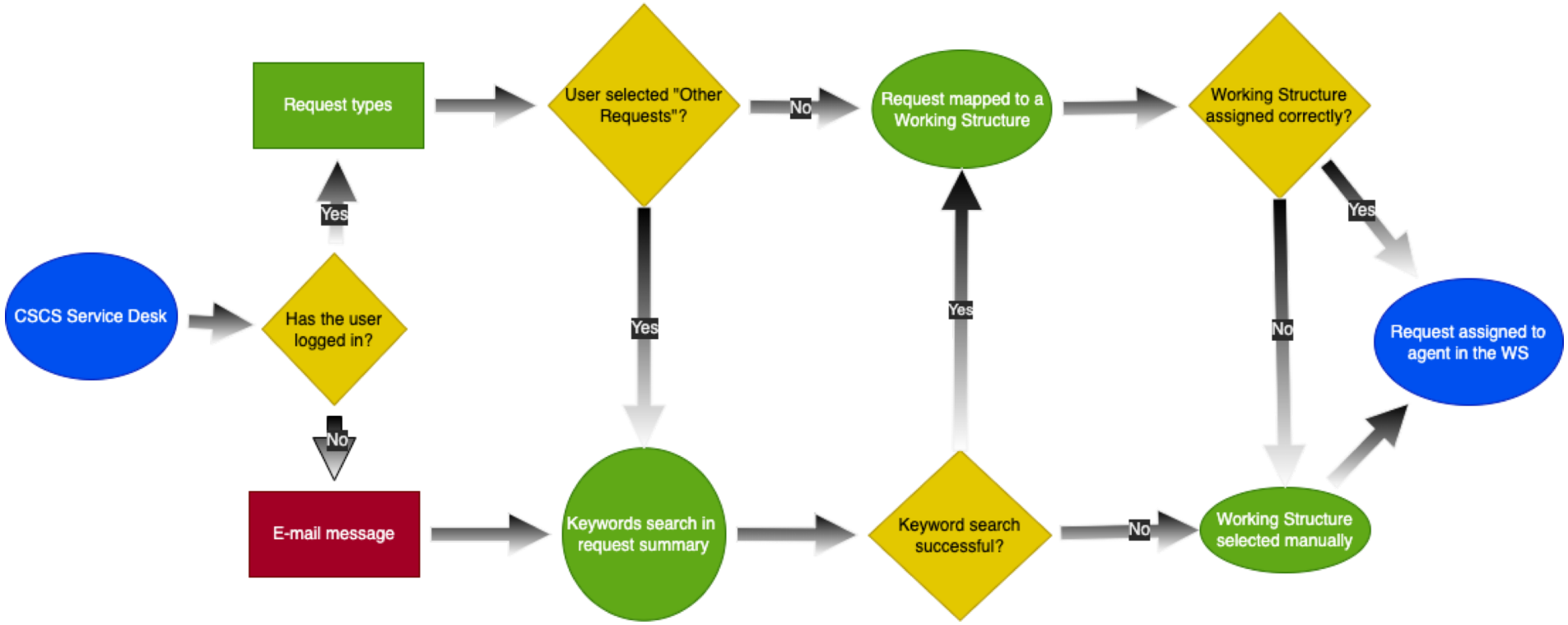
A **timely reaction** by WS is crucial to keep the process efficient

- Periodic **reminders** to are defined by automation rules
- WS and assignees are **notified** based on the status of the request

Requests workflow

No direct WS map of type **Other Requests** and **E-mails**

- Rule-based classification maps these requests if matching keyword is found
- Limited sensitivity and specificity might require multiple iterations



Best practices

Requests should always be **assigned** to a member of a WS

- Avoid public comments in unassigned requests
- Reduce overlapping comments on requests in progress

Requests should be **addressed timely**

- Progress is expected to start within a business day
- Users get notified of request status changes

The **assignee is the primary contact**

- Contributions by colleagues are welcome as internal comments
- Assignees can get help from watchers if needed

Effective flow of user requests

New requests

- Automatic transition to **In Progress with a** public comment
- Otherwise automation rules will send a daily reminder

Requests **Waiting for support**

- The assignee is reminded to provide the requested feedback

Requests **Waiting customer**

- They get automatically resolved without user replies in 14 days

Requests **In Progress** not updated in two weeks (sprint)

- The assignee is automatically reminded to update the status

Monitoring support requests: CSCS Staff

JSM offers multiple ways to monitor efficiently user requests assigned to WS

- **Filters**: save issue searches a filters
- **Dashboards**: create personal dashboards with gadgets based on filters

- Personal subscriptions (CSCS custom implementation)
  - Receive **personal e-mail notifications** of the issues assigned a WS
  - Custom notifications for fields **Project** and **System** available

- **Slack channel notifications**
  - The WS in charge receives periodic reminders
  - Automation rules also send emails to assignees

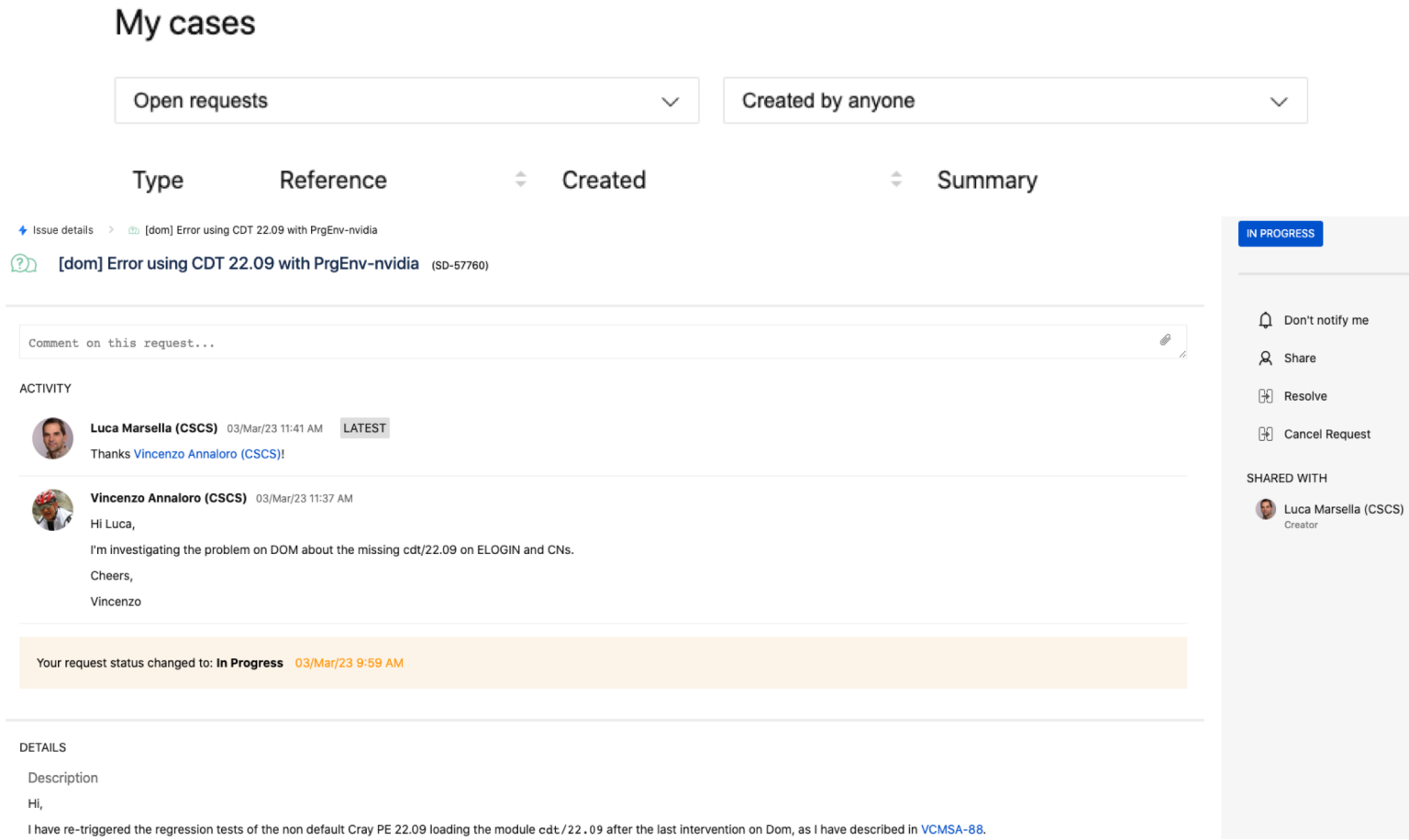
Monitoring support requests: CSCS Users

Cases in the Service Desk:

- **Filter** the list of cases
- Check case **status**
- **Review** the messages

Select a specific case:

- **Share** with other users
- **Resolve** the case if solved
- **Cancel** if sent by mistake



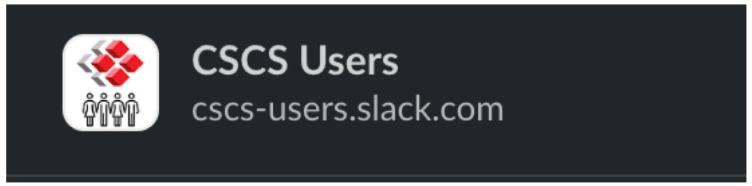
Community Slack Space and Status Page

Community Slack Space

CSCS Users Slack space featuring several channels

A space **to interact with fellow CSCS users**

- **Community** driven approach
- Get **rapid responses** to quick questions
- **Share experiences** and best practices



The Slack space **is not** meant for submitting support requests

- Users submit requests on the CSCS Service Desk
- CSCS staff members are invited to participate
- Replies to inquiries might come from other users

Status Page powered by OneUptime

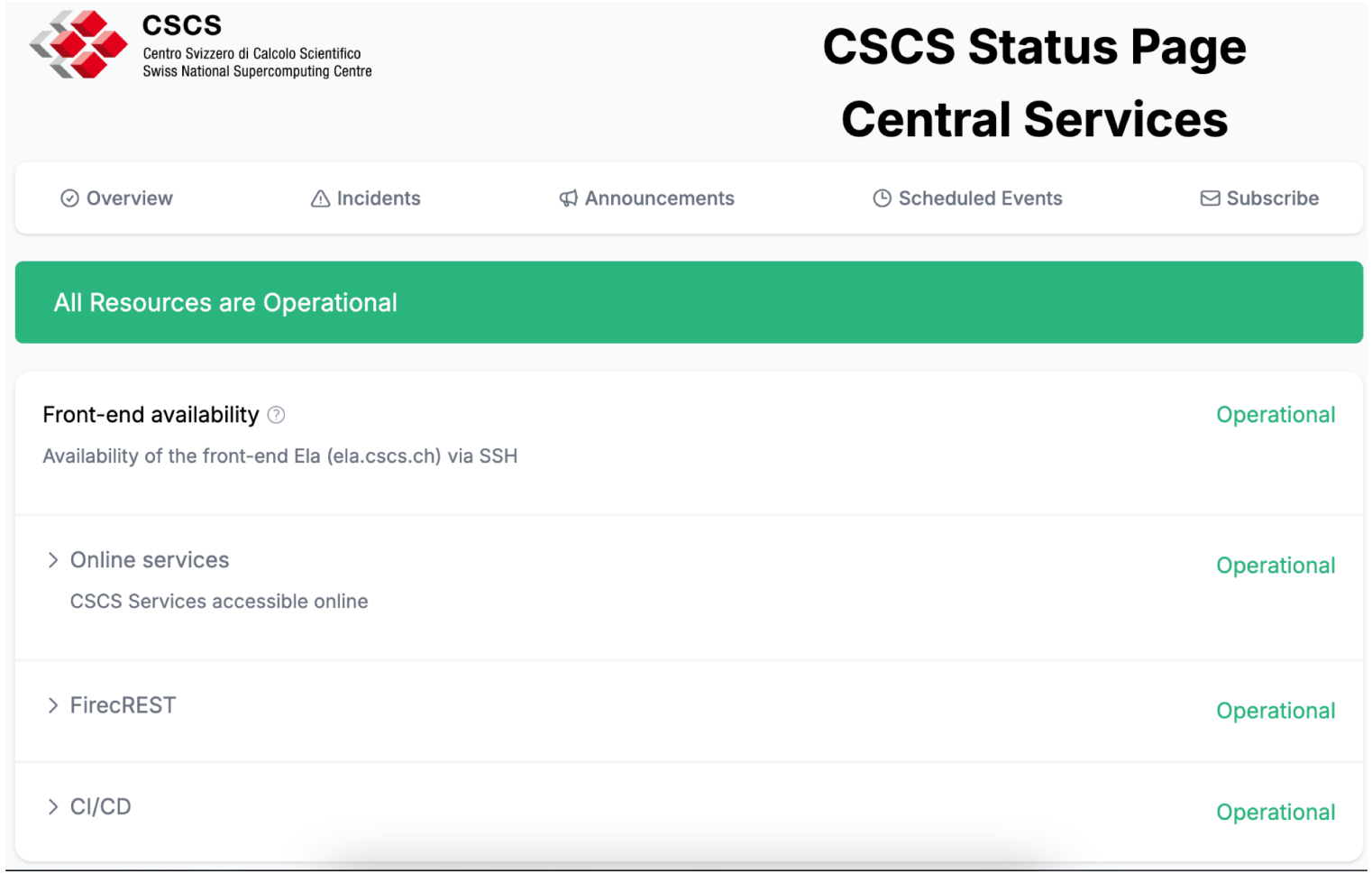
The Status Page reports

- **Incidents**
- **Announcements**
- **Scheduled Events**

Status of resources

- **Online services**
- Front end system
- ….

Subscribe to get notifications



Conclusions

User support services on HPC systems become increasingly complex

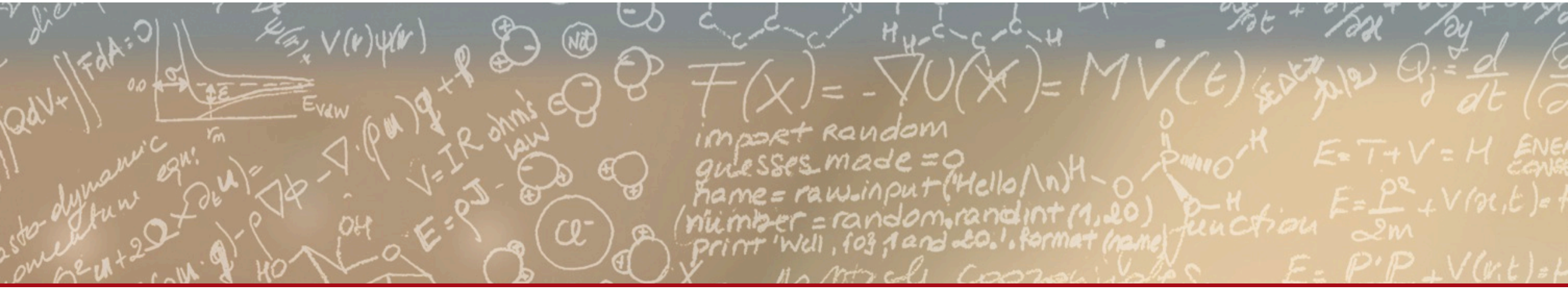
- Technical staff should **focus on project work**
- Automation is implemented to **reduce repetitive tasks**
- Work management tools and **status page** for user communication

Distributed user support system to automate tasks

- Functional structures (WS) grouping staff by expertise
- WS assess and review requests with **end-to-end responsibility**
- Eliminate toil, improve response times, monitor requests traffic

Artificial intelligence and machine learning can ease the burden on staff

- Improve knowledge base from documentation and user requests
- **Chatbots** might help reduce the number of user requests



Thank you for your kind attention