

Rapport

www.infosupport.com

Project Report

Improving Azure Pipelines DX: A Smarter DevOps Experience

Serggio Pizzella

6 januari 2025

Concept

Info Support B.V.

Kruisboog 42
3905 TG Veenendaal (NL)
www.infosupport.com

K.v.K 3013 5370
BTW NL8062.30.277B01
IBAN NL92 RABO 0305 9528 89
BIC RABONL2U

Hoofdkantoor

Tel. +31(0)318 - 55 20 20
info.nl@infosupport.com

Kenniscentrum

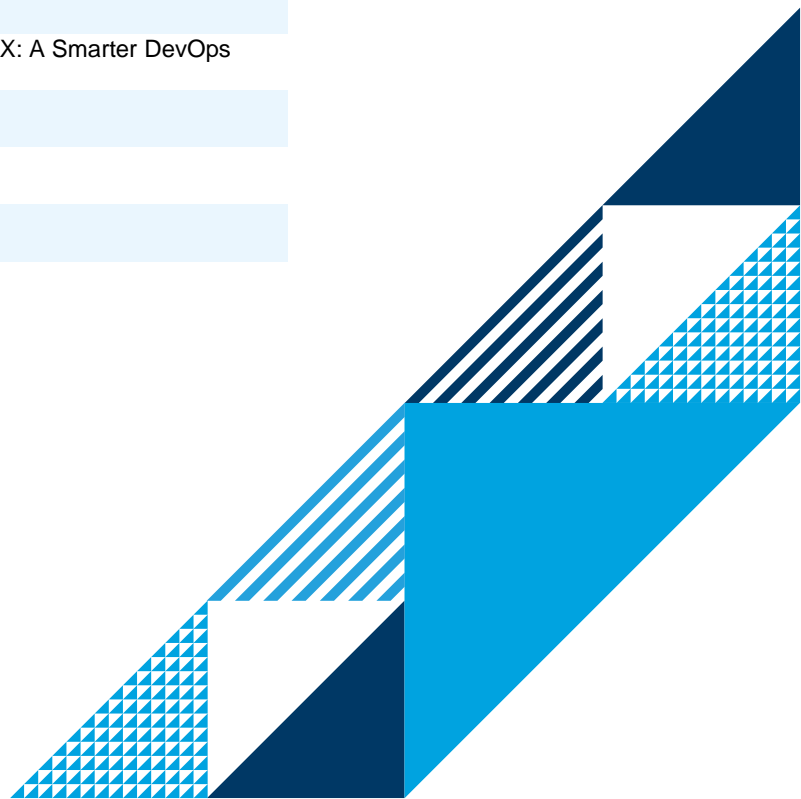
Tel. +31(0)318 - 50 11 19
training.nl@infosupport.com

Project Report

Improving Azure Pipelines DX: A Smarter DevOps Experience

Details

Title	Project Report
Project	Improving Azure Pipelines DX: A Smarter DevOps Experience
Version	1.0
Status	Concept
Date	6 januari 2025





GRADUATION-INTERNSHIP PORTFOLIO BACHELOR-ICT

FONTYS UNIVERSITY OF APPLIED SCIENCES

Student:	
Family name , initials:	Pizzella Ricci, S.V.
Student number:	4449681
project period: (from – till)	02-09-2024 – 03-02-2025
Company:	
Name company/institution:	Info Support
Department:	Business Unit Industry
Address:	Kruisboog 42, 3905 TG Veenendaal
Company mentor 1:	
Family name, initials:	Thissen, N.
Position:	Afstudeer Coodinator
Company mentor 2:	
Family name, initials:	Horsman, L.
Position:	IT Consultant
University teacher:	
Family name , initials:	Schriek, E.
Final portfolio:	
Title:	Improving Azure Pipelines DX: A Smarter DevOps Experience
Date:	10-01-2025

Approved and signed by the company mentor:

Date:

Signature:

History

Versie	Status	Datum	Auteur	Wijziging
1.0	Concept			Creatie
1.1				

Distributielijst

Versie	Status	Datum	Aan
1.0			
1.1			

© Info Support B.V., Veenendaal 2024

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook, zonder voorafgaande toestemming van **Info Support B.V.**

No part of this publication may be reproduced in any form by print, photo print, microfilm or any other means without written permission by **Info Support B.V.**

Prijsopgaven en leveringen geschieden volgens de Algemene Voorwaarden van **Info Support B.V.** gedeponeerd bij de K.v.K. te Utrecht onder nr. 30135370. Een exemplaar zenden wij u op uw verzoek per omgaande kosteloos toe.

Content

1.	Terms	5
2.	Context	6
3.	Reading Guide	9
4.	Additional evidence	12
5.	Process	13
6.	Conclusions	14
7.	Next steps & recommendations	15
8.	Reflection	16
9.	References	17
Bijlage 1	[Titel bijlage]	19



1. Terms

2. Context

2.1 Company

Info Support is a leading consultancy agency founded in 1986. Info Support provides consultancy services and custom-made software solutions to large, well-known companies such as OVPay, Albert Heijn, Jumbo, and Enexis. The company operates internationally, with over 800 employees across five companies. Info Support works across five key sectors: Agriculture & Food, Finance, Healthcare & Insurance, Industry, and Mobility & Public. In addition to its consultancy and software development services, Info Support also focuses on training both internal and external personnel. The company offers certification training in seven fields and provides specialized minors for students, helping them gain expertise during their education.



2.2 Problem statement

Developers often struggle with identifying and resolving errors in Azure DevOps CI/CD pipelines due to the lack of robust local validation tools for the YAML files defining these pipelines. Currently, the only method for validation is committing and testing changes directly in the Azure DevOps environment, which is time-consuming and disrupts workflow efficiency.

This absence of local validation creates repetitive and slow iterations, requiring developers to submit changes to version control, await feedback from CI runs, and address errors. The situation becomes even more challenging when pipelines incorporate multiple shared templates, each introducing unique parameters and considerations.

Common errors such as typos, missing or incorrect parameters, and misuse of variables exacerbate the problem. Each template can introduce its own variables, including nested ones, making the environment increasingly complex and unmanageable. Developers often face long wait times for CI jobs to run, slowing down even minor corrections, such as fixing a typo. This process: saving the file, committing changes with a message, pushing to the repository, waiting for CI to pick up the changes, and running the pipeline to locate the error; can take several minutes for each iteration. Consequently, trivial issues can result in significant delays, productivity losses and developer frustration.

Improving this workflow with real-time, in-editor feedback would drastically reduce these inefficiencies, enhance developer productivity, and reduce frustration.


2.3 Research

Throughout the assignment research will be conducted using the DOT framework. We primarily wish to create a tool find out what the main pain points are in the development process of Azure pipelines, and how can using static analysis. In other words, how can we best alleviate frustration without running the pipeline.

The main question for the project will be:



How can static analysis be applied locally to Azure DevOps pipeline files to maximize mistake prevention before pipeline execution?

From this question stem the following sub-questions:

1. How do Azure DevOps pipeline files function, what are their key components and what are the processes that result in their execution? ( [Literature study](#))
 - a. Dive through the Azure DevOps official documentation to understand the syntax and features available.
 - b. Explore how Azure DevOps parses and executes pipeline YAML files.


Outcome: Flowcharts or diagrams to visually represent the execution flow of a pipeline.

Initially this research was to be conducted before starting the development of the project, however this did not fit the agile methodology chosen for the project. Instead this research was conducted as required by the particular feature that was being worked on.

2. What are the common mistakes and errors occurring in Azure DevOps pipelines, and how can they be identified? ( [Problem analysis](#),  [Root cause analysis](#))
 - a. Gather internal documentation, incident reports, and feedback from developers at Info Support or external to identify common errors.
 - b. Analyse patterns of mistakes.

Outcome: A list of errors, ranked by the impact they would have if resolved. The impact would be informed by the frequency of the error among frequent developers. This list will inform the next stages of development.

This research went beyond providing a list of errors and assembled a comprehensive set of recommendations along with their impact.

3. Which static analysis techniques and tools are best suited for detecting mistakes in Azure DevOps pipelines? ( [Available product analysis: Choosing fitting technology](#))
 - a. Evaluate existing static analysis tools, techniques, and methods (e.g., linters, syntax checkers)
 - b. Assess the feasibility of incorporating or adapting these techniques into the tool being developed.

Outcome: A clear decision on which analysis techniques will be used and what custom rules need to be implemented.

As opposed to writing a single research document, separate ADR's were written as new techniques or technologies were introduced in the project.

4. How can a static analysis tool be developed to integrate into development workflows while ensuring high performance and compliance with internal guidelines? (🔗 [Prototyping](#), 📄 [Non-functional test](#), 📄 [Unit test](#), 🗣️ [Product Review](#))
- Develop the prototype of the static analysis tool, incorporating selected techniques and custom validation rules.
 - Ensure the tool integrates well into existing toolchains and adheres to company guidelines.
 - Keep performance in mind, as to aim for below 1 second validation.

Outcome: A working prototype of the static analysis tool tailored to Info Support's needs.

Not sure if I'm keeping this one.

2.4 [Approach](#) (Local: [Approach.md](#))

The project follows an agile methodology using [Kanban](#) with sprints. Kanban is well-suited for individual work due to its flexibility and simplicity. It allows for continuous task management and adaptation to changing priorities without the need for formal iterations or roles, which would mostly be taken by the student. Throughout the project, the approach shifted to emphasize **vertical slices** within sprints, delivering incremental functionality.

Initially, the project followed a phased plan that separated research, development, and documentation. However, this structure led to challenges in managing concurrent progress, creating a chicken-and-egg problem: in some cases, it was unclear what to develop without first knowing which techniques to use, while in other cases, it was difficult to determine which techniques to research without understanding the requirements of the implementation. This circular dependency caused delays. By Week 12, the approach was revised to integrate research and development into sprints, allowing for iterative progress and addressing techniques as needed for each feature.

By Week 15, the plan was further refined to focus on **template parameters** and **variables**, reducing scope to accommodate the remaining timeframe while ensuring the delivery of core functionalities. A contingency sprint (Sprint 5) was added to allow further coding efforts after the documentation deadline, targeting features that were deprioritized earlier.

The iterative evolution of the approach enabled the project to remain aligned with its goals despite setbacks, delivering a functional tool and comprehensive documentation while maintaining flexibility for future enhancements.

A more extensive description of the approach can be found in the [portfolio site](#).

3. Reading Guide

This section links to the professional products, and describes which learning outcome they contribute to and why.

3.1 Project plan (local)

Learning Outcomes Addressed:

- Professional Duties
- Future-Oriented Organisation
- Investigative Problem Solving
- Personal Leadership
- Targeted Interaction

Initiating and Defining the Project Scope:

At the start of the project, I recognized gaps in my understanding of the intent and requirements. This led me to proactively engage with stakeholders, including Romijn M. and Brandt J., and arrange a meeting with my company mentor and Romijn M. Through this initiative, I clarified the project scope and refined the expectations. The notes for this meeting can be found [here](#). This demonstrates **Targeted Interaction** through stakeholder engagement and **Personal Leadership** in taking ownership of unclear requirements and clarifying them.

Structured Research Approach:

I aligned my research activities with the HBO-ICT research framework, selecting appropriate patterns and methods for each sub-question. This structured approach reflects **Future-Oriented Organisation** by ensuring a methodical and sustainable research process, and **Investigative Problem Solving** through evaluation of research methods best suited to the question.

Iterative Development and Feedback Application:

The project plan underwent four iterations. For each, I actively sought feedback from stakeholders and integrated their suggestions to improve the plan. While the final version was delivered within the available time constraints, I noted areas for future improvement, showing my ability to balance quality with deadlines. This iterative process showcases **Professional Duties** through delivering professional-grade documentation, **Targeted Interaction** by effectively communicating and integrating feedback, and **Personal Leadership** in evaluating and accepting constructive criticism.

3.2 User Requirements Specification (Local)

Learning Outcomes Addressed:

- Professional Duties
- Situation orientation
- Future-Oriented Organisation

Building on Quality Requirements:

The user requirement specification formalizes the functional (FR) and non-functional requirements (NFR) identified during the project planning phase. By adopting user stories and acceptance criteria, I ensured the requirements were both specific and measurable. This demonstrates **Professional Duties** by creating a professional-grade deliverable, essential for guiding development.

Use of Known Techniques:

I utilized established techniques like user stories and acceptance criteria to structure requirements in a way that ensures clarity and aligns with best practices. This approach reflects my ability to apply known methodologies effectively in new contexts, showcasing **Situation-Oriented** skills.

Incorporating Stakeholder Feedback:

Through regular demonstrations and discussions with my company mentor, I refined and expanded the requirements and acceptance criteria based on their input. For example, adjustments were made to ensure alignment with users expectations; multiple fallbacks were determined for the positioning of diagnostics. This iterative collaboration exemplifies **Targeted Interaction**, where communication with stakeholders helped refine requirements and ensure they align with user's expectations.

Formalizing Requirements:

The specification was structured to include:

- **Non-Functional Requirements:** Constraints and quality attributes (e.g., coverage, performance).
- **Functional Requirements:** Clearly defined user stories that outline the desired behavior..

Agile Workflow Integration:

Once defined, the user stories were included in the agile board, integrating them with the sprint-based development approach. This step highlights my ability to bridge planning with execution while ensuring visibility and alignment with the chosen approach.

3.3 Software Architecture (local)

Learning Outcomes Addressed:

- Professional Duties
- Situation orientation
- Future-Oriented Organisation
- Investigative Problem Solving
- Targeted Interaction

Architectural Decisions and Documentation:

Throughout the project, I documented big architectural decisions using the MADR (Markdown Architectural Decision Records) template, as detailed in the ADRs. This approach ensured a clear and defensible rationale for each decision, exemplifying **Professional Duties** and **Investigative Problem Solving** by applying industry-recognized practices and conducting thorough evaluations of alternatives.

Use of Standardized Models:

The architecture was visualized using the C4 model, supported by sequence diagrams and additional diagrams where needed. This adherence to industry standards ensured that the design was clear, communicable, and easily understandable for stakeholders, reflecting **Professional Duties** and **Situation Orientation** by adapting to the audiences' expectations.

Guiding Principles:

The architecture design was guided by the project's non-functional requirements (e.g., IDE independence, seamless integration, real-time validation). These constraints were derived from the User Requirements Specification and aligned with stakeholder needs, showcasing **Future-Oriented Organisation** by considering scalability and maintainability.

Iterative Updates:

The architecture evolved alongside the project, with incremental updates to reflect changes in scope and features. This continuous alignment with the project illustrates **Future-Oriented Organisation** ensuring that the architecture remained relevant and effective.

Stakeholder Collaboration:

I regularly presented architectural decisions and diagrams to stakeholders for feedback, adapting designs based on their input. The initial idea to add future extensions within the diagrams, came from them. This demonstrates **Situation Orientation** and **Targeted Interaction**, ensuring the architecture aligned with stakeholder expectations and projects' goals.

4. Additional evidence

5. Process

6. Conclusions

7. Next steps & recommendations

8. Reflection

9. References

Bijlagen

Bijlage 1 [Titel bijlage]

Klik of tik om tekst in te voeren.

