www.infosupport.com

Project Plan

Local Validation Tool for Azure DevOps Pipelines

Auteur

6 September 2024

Proposal Final

Info Support
Solid Innovator

Hoofdkantoor
Kruisboog 42
3905 TG Veenendaal
Nederland
Tel. +31(0)318 - 55 20 20
Fax +31(0)318 - 55 23 55

Kenniscentrum
De Smalle Zijde 39
3903 LM Veenendaal
Tel. +31(0)318 - 50 11 19
Fax +31(0)318 - 51 83 59

info.nl@infosupport.com
www.infosupport.com

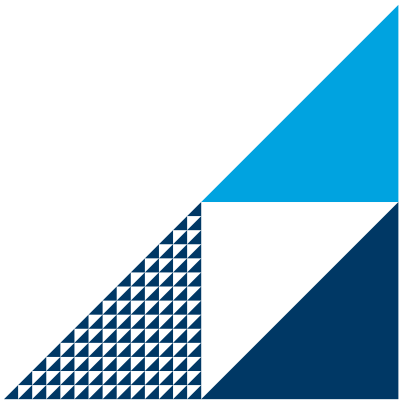
K.v.K 3013 5370
BTW NL8062.30.277B01
IBAN NL92 RABO 0305 9528 89
BIC RABONL2U

IBAN NL74 INGB 0004 7385 93
BIC INGBNL2A

Project Plan

Local Validation Tool for Azure DevOps Pipelines

Titel	Project Plan
Project	Local Validation Tool for Azure DevOps Pipelines
Version	1.0
Status	Proposal Final
Date	6 September 2024
Company	Info Support

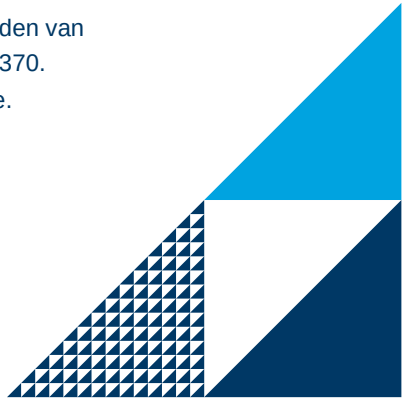


History

Version	Status	Date	Auteur	Change
1.0	Concept	06/09/2024	Serggio Pizzella	Creation
1.1	Concept	10/09/2024	Serggio Pizzella	“Company”, “Graduation”, “TRL”
1.2	Proposal Final	18/09/2024	Serggio Pizzella	“Research”, Rework TRL, Add keyword definitions, “Approach”
1.3	Proposal Final	25/09/2024	Serggio Pizzella	Split approach by research question, apply Nikko’s feedback.
1.4	Final	27/09/2024	Serggio Pizzella	Expand to not only template errors. Minor readability tweaks.

Distribution list

Version	Status	Date	To	Means
1.0	Concept		Luuk Horsman	Teams
1.1	Concept	10/09/2024	Erik van der Schriek	Teams
1.2	Proposal Final	18/09/2024	All Stakeholders	Canvas Submission & Teams
1.3	Proposal Final	25/09/2024	All Stakeholders	Canvas Submission & Teams
1.4	Final	27/09/2024	All Stakeholders	Canvas Submission & Teams



Contents

1. Company

This graduation internship will be conducted at Info Support, 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. Assignment

2.1 Stakeholders

Role	Name	Contact
Graduate student	Serggio Pizzella	serggiopizzella@gmail.com
Company Supervisor	Luuk Horsman	Luuk.horsman@infosupport.com
Fontys 1st assessor	Erik van der Schriek	e.vanderschriek@fontys.nl
Fontys 2nd assessor	Nico Kuijpers	nico.kuijpers@fontys.nl

2.2 Assignment Definition

Developers frequently encounter challenges when identifying and resolving errors within CI/CD[1] pipelines in Azure DevOps, specifically due to the lack of local validation tools for the YAML files that define the pipeline. Currently, the only way to validate YAML pipelines is to commit and test them directly in the Azure DevOps CI/CD environment, a process that slows down the development workflow.

2.2.1 Problem Statement

The lack of local validation for YAML files in Azure DevOps pipelines leads to time-consuming iterations, as developers need to submit their changes to version control, await feedback, and correct errors. This issue becomes more complex when pipelines involve multiple templates shared across teams, each requiring their own considerations.

Core technologies like YAML, the Azure Pipelines schema, and the Language Server Protocol (LSP) are mature and have already reached TRL[2] 9. However, the innovation in this project lies in combining these well-established technologies to create a tool that addresses a specific gap—local pipeline validation within the Azure DevOps ecosystem.

The project begins at TRL 4 due to the maturity of the enabling technologies, with a focus on building a small-scale prototype that integrates these components. The tool will provide local validation of Azure DevOps pipelines, detect an approved set of errors, and offer real-time feedback within a controlled environment.

The explorative elements of the project include developing the validation library, a CLI, and an LSP integration, as these have not been previously implemented for this specific use case.

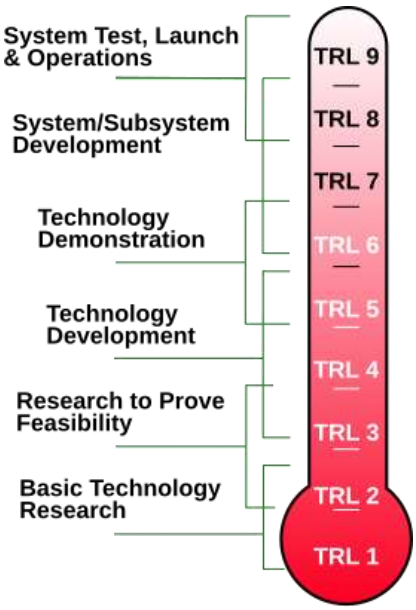


Figure 1: TRL scale

[1] Continuous integration & Continuous delivery
[2] Technology readiness level

2.2.2 Research

Throughout the assignment research will be conducted using the DOT framework. We primarily wish to find out what the main pain points are in the development process of Azure pipelines, and how we can alleviate them 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 techniques 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.

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.

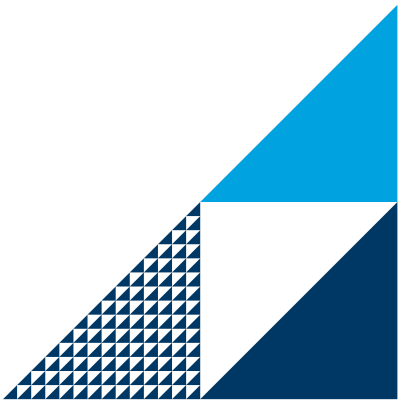
   Outcome: A list of errors, ranked by the impact they would have if resolved. The impact would be informed by the frequency of the error among frequent developers. This list will inform the next stages of development.

3. Which static analysis techniques and tools are best suited for detecting mistakes in Azure DevOps pipelines? (Available product analysis)
   - **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.

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)
   - **a.** Develop the prototype of the static analysis tool, incorporating selected techniques and custom validation rules.
   - **b.** Ensure the tool integrates well into existing toolchains and adheres to company guidelines.
   - **c.** 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.



2.2.3 The tool

The main goal of the assignment is to develop a tool that allows proactive validation of pipelines locally, before committing them to Azure DevOps. The solution should reduce errors in pipelines and enhance overall development efficiency by minimizing iteration time.

The tool developed as part of this project will be divided into three components:

1. **Validation Library**:
   The core of the tool will be a validation library responsible for performing the actual validation of Azure DevOps pipelines. This library will handle parsing and checking .yaml files, ensuring that template parameters and variables are correctly validated. It will be designed to be modular and flexible, allowing for future enhancements or integration with other systems.

2. **Language Server Protocol (LSP) Integration**:
   An additional thin wrapper will be implemented around the validation library to integrate with the Language Server Protocol (LSP). This will allow the tool to provide real-time feedback within various Integrated Development Environments (IDEs) that support LSP, such as Visual Studio Code. The LSP integration would make the tool even more accessible, providing inline error detection and corrections directly within the developer's coding environment.

3. **CLI:**
   A thin wrapper will be built around the validation library to provide a command-line interface (CLI). This will allow developers to easily run validations locally from the terminal without the need for a complex setup. The CLI will offer simple commands to validate pipelines and display results in a user-friendly format, ensuring quick feedback for the user.

By dividing the tool into these three components, the project ensures flexibility and ease of use, while catering to different user preferences and workflows. The modularity of the system also allows for potential future extensions or integrations with other tools.

2.2.4 Graduation

The final part of the assignment will focus on the graduation of the student. This includes fulfilling the learning outcomes required by Fontys University:

- **Professional Duties**: Carrying out professional tasks at a bachelor level, resulting in professional products in line with the IT area of the project.
- **Situation-Orientation**: Applying previously acquired knowledge and skills in a new, real-world context to deliver valuable results for both the project and the company.
- **Future-Oriented Organisation**: Exploring the organizational context of the project, considering business, sustainability, and ethical aspects, while managing the project execution.
- **Investigative Problem Solving**: Critically evaluating the project, identifying problems, and finding effective approaches to solve them.
- **Personal Leadership**: Demonstrating an entrepreneurial attitude and focusing on personal growth, with attention to future career goals in the IT field.
- **Targeted Interaction**: Identifying relevant stakeholders, collaborating constructively with them, and communicating effectively to achieve project success.

In addition to meeting these outcomes, this part of the assignment includes the creation of a portfolio for graduation, preparation for the defence of the graduation internship. The project must meet the expectations of both Info Support as the host company and Fontys University as the academic institution.

Additionally, the student is working towards resolving 6 missing ECTS, which is required to complete the degree.

2.2.5 Result:

The goal of this project is to build a tool that solves the issue of not being able to locally validate YAML pipeline files in Azure DevOps. By the end of the project, the tool should:

- Validate pipeline files locally, allowing developers to catch errors before they commit changes.
- Identify and fix discovered errors in CI/CD pipelines, making the development process smoother and more reliable.

In addition to the tool itself, the following deliverables are required:

- Documentation: The project will include technical documentation in the form of C4 diagrams to illustrate the system architecture and how the components interact.
- Unit Tests: Comprehensive testing is critical, with a goal of achieving 80%-unit test coverage to ensure the tool's reliability and robustness.
- Quality Control: The code quality will be tracked using SonarQube or other quality control metrics to ensure best practices and maintainability.
- CI/CD: Continuous Integration (CI) will be implemented as part of the development pipeline, while Continuous Deployment (CD) is not a strict requirement for this project.
- Internal Guidelines: The tool's development will adhere to internal company guidelines, as outlined on guidance.infosupport.nl.

The tool will use established technologies like YAML, the Azure Pipelines schema, and LSP, but the new part of this project is how these will be combined to allow local validation. The project aims to reach **TRL 6**, meaning the tool will have been tested and used against real pipeline files made at the company.

2.3 Priorities

In case time constraints come up and decisions need to be made on which part of the project to focus on, priority will always be put first on the Graduation aspects of the project, with further priority on obtaining the 6 missing ECTS, as without them the project is bound to be cancelled. The next priority will be put on the `validation library` component of the tool, followed by the LSP wrapper. The least priority going to the cli wrapper component.

2.4 Scope

Included:

- Local validation of .yaml files for Azure DevOps pipelines.
- Validation of template parameters and variables.

Excluded:

- Checking the correctness of the pipeline's internal logic, such as task dependencies or execution order.
- Validation of files to which the user has no access to.

2.5 Dependencies

2.5.1 Failure to Obtain Missing 6 ECTS

If the student is unable to obtain 6 ECTS — either due to not finding a suitable course or failing the required course — the Examination Board has the authority to cancel the graduation project. This would result in a delay or potential inability to graduate for the student.

2.5.2 Microsoft Implements the Tool

There is a possibility that Microsoft may develop and release an official tool that enables local validation of Azure DevOps YAML pipelines during this project. If this occurs, it could impact the relevance and necessity of the student's tool, posing a significant dependency risk.

If Microsoft releases such a tool:

- **Project Adaptation:** The project scope may need to be adjusted to focus on areas not addressed by Microsoft's tool. This could involve enhancing the tool with additional features, improving usability, or targeting niche use cases.
- **Stakeholder Consultation:** The student must promptly inform the company supervisor and academic assessors. Together, they can decide on the best course of action, which may include:
    - **Reframing the Project:** Shifting the project's focus to complement Microsoft's tool or to address new problems identified.
    - **Comparative Analysis:** Turning the project into a comparative study between the developed tool and Microsoft's solution, highlighting differences, advantages, and potential improvements.
    - **Alternative Solutions:** Exploring other related issues within the Azure DevOps ecosystem that require attention.

2.6 Quality requirements

The success of this project depends on two main quality requirements: user-friendliness and performance. These are required to ensure that the tool is adopted by developers and adds value to the CI/CD pipeline development process.

2.6.1 User-Friendliness

The tool must be easy to set up and integrate into the developer's existing workflow. Ideally, the tool should require no setup steps or configuration, aside from credentials, allowing developers to use it immediately after installation. Any necessary configuration steps must be minimal and automated where possible. The goal is to make the tool as seamless as possible so that it fits naturally into a developer's local development environment without causing friction.

2.6.2 Performance

The tool must provide validation results quickly. If the tool takes longer to validate pipelines than the current workflow of committing to Azure DevOps and waiting for feedback, it will defeat the purpose of reducing iteration time. Validation should occur in real time or with minimal delay to ensure the tool improves efficiency.

2.7 Predefined conditions

The following decisions and conditions have been set by the company supervisor and will guide the development of the tool:

- **IDE Independence**
  The tool must function regardless of the Integrated Development Environment (IDE) being used by developers. Whether the user is working in Visual Studio Code, JetBrains, or any other IDE, the tool must operate effectively without requiring IDE-specific configurations or dependencies.
- **Sub-second response time**
  The tool must provide diagnostics within 1 second, after startup. Startup includes, among others, the fetching of required files, such as template files.

3. Approach

3.1 Methodology

For this project, Kanban (with sprints) will be used as the agile methodology. 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.

3.2 Phases:

3.2.1 Planning Phase (Weeks 1-4):

- **Objective:** Define project goals, scope, and requirements.
- **Deliverables:** Project plan.

3.2.2 Research sub-question 1 & 2 (Week 4-6):

- **Objective:** Gain insight into how azure pipelines work and leverage that insight to find the pain points.
- **Deliverables:** Execution flow document + diagrams, Ranked list of errors.

3.2.3 Research sub-question 3 (Week 7-9):

- **Objective:** Determine which methods or techniques can be used to solve the pain points previously discovered; rank the ease of implementation and cross-reference them with their impact to determine which solutions will be implemented.
- **Deliverables:** Ranked list of techniques to implement; an approved subset for implementation; and documents detailing how the technique works and may be implemented.

3.2.4 Midway checkpoint (Week 10)

- **Objective:** Receive indicative feedback on the progress of the project, by presenting the progress thus far.
- **Deliverables:** Presentation of all the deliverables thus far and gained insights; Midterm evaluation by company mentor.

3.2.5 Research sub-question 4, Development and Testing Phase (Weeks 11-18):

- **Objective:** Implement features and components of the tool through iterative 2-week sprints. Testing is integrated into each task as part of its completion, ensuring quality and performance standards are met. Utilize a hybrid methodology combining elements of Kanban and Scrum to manage workflow effectively.
- **Initial week:** setup a task backlog.
- **Approach:**
    - **2-Week Sprints:** The project will be organized into 2-week sprints, each beginning with planning and ending with a review.
        - **Sprint Planning:** At the start of each sprint, define the goals, tasks, and priorities based on the project backlog.
        - **Sprint Research:** Perform the required research for the sprint's goals.
        - **Sprint Review:** At the end of each sprint, assess completed work, gather feedback, and adjust the backlog as needed.
    - **Integrated Testing:** Due to the complexity of the project and the high code coverage goal, integrating testing into each sprint is crucial. To assert the effectiveness of the tool, real pipelines files will be used.
- **Feedback:** Additional feedback may be gathered throughout a sprint.
- **Deliverables:**
    - **Validation Library:** Development and testing of the core validation library.
    - **CLI Interface:** Development and testing of the command-line interface wrapper.
    - **LSP Integration:** Initial development and testing of the Language Server Protocol integration.
    - **Updated Documentation:** Continuous updates to technical documentation and research documents.

3.2.6 Documentation and Finalization Phase (Weeks 18-19):

- **Objective:** Complete all necessary documentation and prepare for the final submission. Ensure all deliverables are polished and meet required standards.
- **Deliverables:** Final project documentation, project report[3], graduation portfolio, and updated project presentation.

3.3 Kanban Board Structure:

The Kanban board will be organized into the following columns to manage tasks effectively:

- **To Do:** Planned tasks awaiting initiation.
- **In Progress:** Tasks currently being worked on.
- **In Testing:** Task implementation completed, working on tests.
- **Review:** Tasks completed and pending review.
- **Done:** Completed tasks that have passed review.

[3] See: https://fhict.instructure.com/courses/13872/pages/portfolio-tips

4. Management Aspects

4.1 Organization

4.1.1 Standards

The project is organized to ensure clear communication, efficient workflow, and effective decision-making. The organizational structure includes the following roles:

- **Graduate Student:** is responsible for the planning, execution, and delivery of the project. This includes development, testing, documentation, and ensuring that the project meets both company and Fontys requirements.
- **Company Supervisor:** acts as the primary liaison between the graduate student and Info Support. He provides guidance, technical expertise, and ensures that the project aligns with the company's expectations and standards.
- **Fontys Assessors:** They oversee the academic integrity of the project, ensuring compliance with the learning outcomes. They provide feedback on academic deliverables and assess the student's performance.

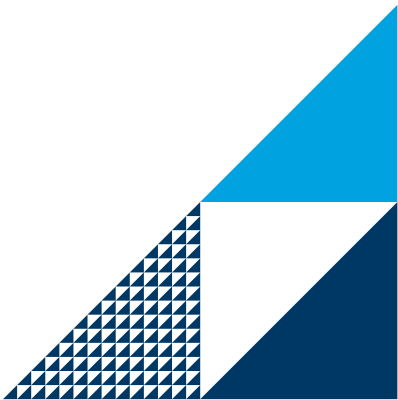
Relationships:

- **Project and Client (Info Support):** Direct communication is maintained between the graduate student and the company supervisor. Regular meetings ensure alignment with company goals and expectations.
- **Project and Academic Institution (Fontys):** The graduate student communicates progress and submits required deliverables to the academic supervisors, ensuring adherence to academic standards.
- **Steering Committee/Advisory Group:** There is no formal steering committee or advisory group. However, feedback from both company and academic supervisors guides the project's direction.

4.1.2 Progress Monitoring

Progress monitoring for organizational aspects includes:

- **Bi-Weekly Meetings with Company Supervisor:** The graduate student will have bi-weekly meetings with *Luuk Horsman* to discuss progress, challenges, and receive guidance.
- **Weekly Reports to Academic Supervisors:** Every week, the graduate student will send progress reports to *Erik van der Schriek*, summarizing achievements and upcoming plans, in the form of checkpoints through FeedPulse.
- **Periodic Evaluation:** The effectiveness of the project organization will be evaluated during progress meetings. Adjustments to roles or communication strategies will be made as necessary.



4.2 Quality

4.2.1 Standards

- **Functional Requirements:**
  - The tool must enable local validation of Azure DevOps YAML pipeline files.
  - It must provide diagnostics about the issues it discovers.
- **Non-Functional Requirements**
  - The tool must be thoroughly tested, with an aim of 80% coverage.
  - The tool must provide result within 1 second.
  - The project must utilize tools like SonarQube to monitor code quality, including metrics such as code complexity, duplications, and adherence to coding rules.
  - The tool must integrate seamlessly into developers' workflows, requiring minimal setup.
  - The codebase must adhere to industry best practices and internal company coding standards.

4.2.2 Standards Monitoring

- **Code Quality Reports:**
  - **SonarQube Analysis:**
    - Regularly run SonarQube scans or equivalent to generate reports on code quality and coverage.
    - Monitor key metrics such as code smells, bugs, vulnerabilities, and duplications.
    - Address any issues identified in the reports.
- **Testing:**
  - **Unit Testing:**
    - Write unit tests for all significant pieces of code.
    - Maintain the goal of achieving at least 80% code coverage.
- **Continuous Integration:**
  - Set up a CI pipeline that automatically runs tests and code quality analysis upon code commits.
  - Ensure that the pipeline includes steps for building the code, running tests, and performing code analysis.
- **Performance Testing:**
  - Conduct performance tests to verify that the tool meets the required speed and efficiency benchmarks.
  - Optimize code where necessary to improve performance.
- **Regular Reviews:**
  - **Code Reviews:**
    - Even as an individual developer, perform thorough self-reviews of code.
    - Optionally, seek peer reviews from colleagues or mentors for critical components.
- **Quality Reporting:**
  - Include quality metrics and progress in meetings with the company supervisor.
  - Discuss any quality concerns or improvements during meetings with the company supervisor.

5. Notes

5.1 Keywords

Keyword	Acronym	Meaning	Source
Continuous integration	CI	The practice of integrating source code changes frequently and ensuring that the integrated codebase is in a workable state.	Wikipedia
Continuous deployment	CD	The practice of frequently delivering functionalities trough automated deployments.	Wikipedia
Technology readiness level	TRL	A method for estimating the maturity of technologies during the acquisition phase of a program	Wikipedia