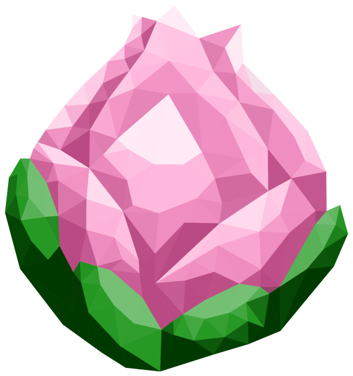
Vision document

Cion

Version 1.0



Revision history

Date	Version	Description	Author
31.01.18	0.1	First draft	Kenan Mahic, Erlend Tobiassen, Harald Wilhelmsen
20.04.18	0.2	Second draft	Kenan Mahic, Erlend Tobiassen, Harald Wilhelmsen
27.05.18	1.0	Final Draft	Kenan Mahic

65

Table of Contents

1 Introduction

The purpose of this documents is to describe the envisioned end-product of our bachelor’s-assignment. As well as to act as a guideline for the design of the requirement document. The scope of this document is cion, its components and utilization. In addition to describing the stakeholder and end users. The document describes the need for cion in relation to both the stakeholder and end-users. It also pans out the differences between cion and our competitors. Furthermore the features in the vision document are described on a higher level, without implementation specifics. This makes the vision document better suited as a read for the layman.

Overview: Describe the vision-document contents and explain how the document is organized.

- License, opensource, etc

1.1 References

2 Positioning

2.1 Problem Statement

Problem	The issue at hand is that current Continuous Deployment tools are neither configurable enough for the issuer's liking, nor do they have the ability to host the tools on their own platforms. As well as being lightweight.
affects	The issue affects at the very least the stakeholder of the assignment, as well as other parties that would like to self-host a lightweight Continuous Deployment tool on their own platforms.
as a result	The issuer can only use the Continuous Delivery paradigm and not the Continuous Deployment one. The situation is that they have to push new updates manually to production for their services.
a successful solution would	automate the pushing of updates to production safely and securely. Saving both the time needed to manually push the updates, and the time it takes to set up a secure enough instance of CD-tools already developed.

2.2 Product position statement

For	Trondheim Kommune
that	needs a secure and self-hosted CD-tool.
cion	is a solution that does exactly that.
that	the most important advantage is it being able to be self-hosted. As well as being highly configurable.
unlike	most of today's competitors, who are cloud services. This entails you having to open up ports in your firewall to give the CD-tools access to updating your services, this raises a lot of security concerns.
our product	lets you host it on your platforms, as well as being more configurable than our current competition

3 Stakeholder and User Descriptions

3.1 Stakeholder Summary

Name	Description	Role under development
Runar Andersstuen	Runar is Trondheim Municipalities representative during the duration of the project. He is a natural stakeholder as he is part of the team that will use the final product when finished	Runar will act as an advisor, giving us tips and criticisms. As well as act on behalf of Trondheim municipality to see to that the requirements are met.

3.2 User Summary

Name	Description	Role under development	Represented by
Simen Bentdal	The user is as of writing a developer for Trondheim municipality. He has an extensive background in systems-engineering and development and is not a layman in terms of Continuous Deployment. This would be the case for all our intended users	Simen will be the primary external tester. He will be used in Use Cases and asked for feedback	Represents himself.
Lars Tore Vassli	-//-		-//-

3.3 User Environment

The final product is meant to be run as part of a CD-solution, where CI is already handled. But can also be run completely without a CI-system in place. As long as the user manages to trigger the cion-catalyst.

In Trondheim Municipality they use Bitbucket Pipelines to build their code and docker images and push them to dockerhub. Dockerhub is then configured with a webhook to send an HTTP POST-request to cion-catalyst, activating the service-update process.

3.4 User Profiles

Culmination of the needs of the users

Need	**Priority**	**Affects**	**Today's Solution**	**Proposed solution**
Kubernetes support	High	Users of kubernetes environments	None	Users will have the option uploading images to a kubernetes environment
Scheduled deployment	High	Trondheim municipality	Deploys manually or at the time of uploading a new image	User will be able to schedule a specific time for deployment
Permission system	High	Everyone	All users have full access to cion' capabilities	An administrator should be able to limit a user' access. For example restrict access for deployment to production.
Post-deploy behavior	High	Everyone	Does not send any webhook/notification of any kind post deployment.	Be able to send a webhook after deployment so that the users can for example run automated tests.

3.5 Alternatives and Competition

3.5.1 Bitbucket pipelines

Bitbucket Pipelines supports automatically running certain commands whenever a build finishes. This means that you as a user of Pipelines can configure a command that updates the services in your swarm. But this requires that you open up your system for remote commands, which is a huge security risk. The reason for this being an issue is that you can not open up your system only towards bitbucket and filtering out everything except service update calls. If you open it up it is open to everything.

To make this solution secure you need a system in-between to receive the commands from Pipelines to validate the source and filtering out certain commands. cion can be exactly this system, receiving this command and validating the source and call before applying the update.

3.5.2 Jenkins and other extensible CI tools

Jenkins is an extensible CI/CD tool that allows the user to install third-party plugins to handle both continuous integration and continuous deployment. Plugins can do a lot more, but the CI/CD ones are what is relevant for this document. It can be hosted locally by the user, but its setup can be quite cumbersome.

3.5.3 cron-jobs and other polling-style setups

When docker is running in in linux it is possible to set up scheduled jobs to run locally on a time interval. So basically a "docker service update --image [...]" command that runs every 5 minutes or so. This does in theory replace the main use-case of cion. In that everytime a new image is available, it would update your services accordingly, but it would supply very minimal logging such as error-reporting and general logging, and it would be tedious to set up new services, due to you having to set up new jobs every time you add a new service to your environment. cion supplies a web GUI for adding new services and environments, and a feature-rich log-traversing tool.

4 Product Overview

4.1 Product Perspective

cion is designed to run as part of a larger CD-solution, but can also be run as alone.

When using it as part of a CD-solution you would configure automatic webhooks on external services like dockerhub, to prompt cion that a new image now exists. cion would then take over and pull the image from dockerhub and update the services that use that image.

But cion does not necessarily have to be triggered by an external source, this is only necessary when the user wants automatic updates of services. cion can also be triggered manually through its web UI, which would make that instance of cion an independent system.

4.2 Assumptions and Dependencies

The basic assumptions made are that the stakeholders want a CD-tool they can host themselves. If this was not the case, there are some competitors that have cloud-solutions that do the same.

We are also inherently dependent on docker and kubernetes. Without docker images and some type of containers our solution would require severe changes. We also presume the images are linux environments, if not we would also have to make major changes.

5 Product Features

Feature	**Description**
Login	Log in through username and password
Logout	Log out of the page, invalidating the session.
Change user avatar	Change the avatar that is bound to your user
Change password on self	Change the password on your own user
Create user	Create a new user with configured permissions, username and password.
Delete user	Delete a user
Post Deploy webhooks	Configurable webhooks to be triggered as services are updated.
Manually trigger a new service-update	Create a service-update task to be completed by cion.
View what image was last deployed to what environment	View what image was last deployed and when it was deployed per service per environment.
View service-update history	View a list of what images have been deployed over all time per service.
Kubernetes Support	Allow users to configure kubernetes services when creating a new service.
Block features behind permissions given to a user	Block features behind a permission-system to allow certain features to be used by certain users.
Configure permissions for user	Let users with the user-edit permission edit the permissions of other users. (typically an

	admin user)
Live view of service-updates happening	A live view of events happening in cion. Typically service-updates or when a new image has been detected for a configured service.
Table Pagination to traverse data	Paginating, searchable and sortable UI-tables on database-tables, for example the tasks-table, that can become quite large.
Registry notification support catalyst	Extending support to local docker registries
Service overview	View services that are configured to be updated to see what services are configured and where they are configured to run.
Configure new service	Add service which updates are to be managed by cion
Environment overview	View environments configured to be managed by cion.
Configure new environment	Add environment to be managed by cion

6 Other Features and Requirements

As the product is meant for expert-users we value configurability more than usability to a certain degree. This means the development of the UI considering aesthetics does not play a hugely significant role for cion. But the end product will need to be well documented, as most of the features might not be the most intuitive. The product will be designed to be lightweight and extensible, and releasing the code as open source helps the latter.

Performance is not a big concern since cion will predominantly perform an update of services' images, in practise this will involve downloading a image file anywhere from 5MB to several GB and then restarting the service. Both of these actions will take orders of magnitude more time than the logic involved. This allows us to use python as our language without suffering from its performance loss compared to e.g. C++.

However reliability is very important to cion, we will have to depend on external webhooks that notifies cion when images get a new version. The risk here is if cion is not available when a image receives an update then we will miss our only chance at processing the webhook. In order to deal with this we require the following features.

Feature	**Description**
Isolated component	Isolate the webhook component so that it can be run independently from the rest of cion.
Separate cache	A cache in the webhook component so that if the database layer is unavailable the webhook is not lost.