Cognite AS
Fornebuporten, Oksenøyveien 10
1366 Lysaker, Norway

NOC MVP#1 SCALE UP PROJECT ARCHITECTURE

Document Title	NOC MVP#1 Scale-Up project architecture
File Name	NOC MVP#1 Scale-Up project architecture v12.0
Current Version	12.0
Change log	• CT14 - Machine Learning and Advanced Analytics Readiness
Date of Current Version	June 21st 2020
Status	Distributed for NOC approval
Security	External

Content: *This document contains a detailed overview of project software architecture split into 14 core topics as defined by NOC*

Intended audience: *NOC IS SMEs and other parties related to MVP#1 Scale Up project software architecture for NOC*

Cognite AS
Fornebuporten, Oksenøyveien 10
1366 Lysaker, Norway

Cognite AS
Fornebuporten, Oksenøyveien 10
1366 Lysaker, Norway

TABLE OF CONTENTS

INTRODUCTION

A robust and scalable architecture is essential to the success of the MVP#1 Scale Up Project. This document is a collaborative effort between NOC and Cognite, and is based on Cognite's best practices for software integration and deployment, and the specific patterns to be applied for MVP#1 Scale-Up delivery according to NOC requirements. The intention is to describe aspects of the architecture design and delivery in detail, and to deliver documentation and insight to established NOC DT architecture principles and core topics. The document is structured around 6 agreed principles (Quality, Scalability, Security, Flexibility, Standardization and Innovativeness), the underlying core topics, and a table of contents.

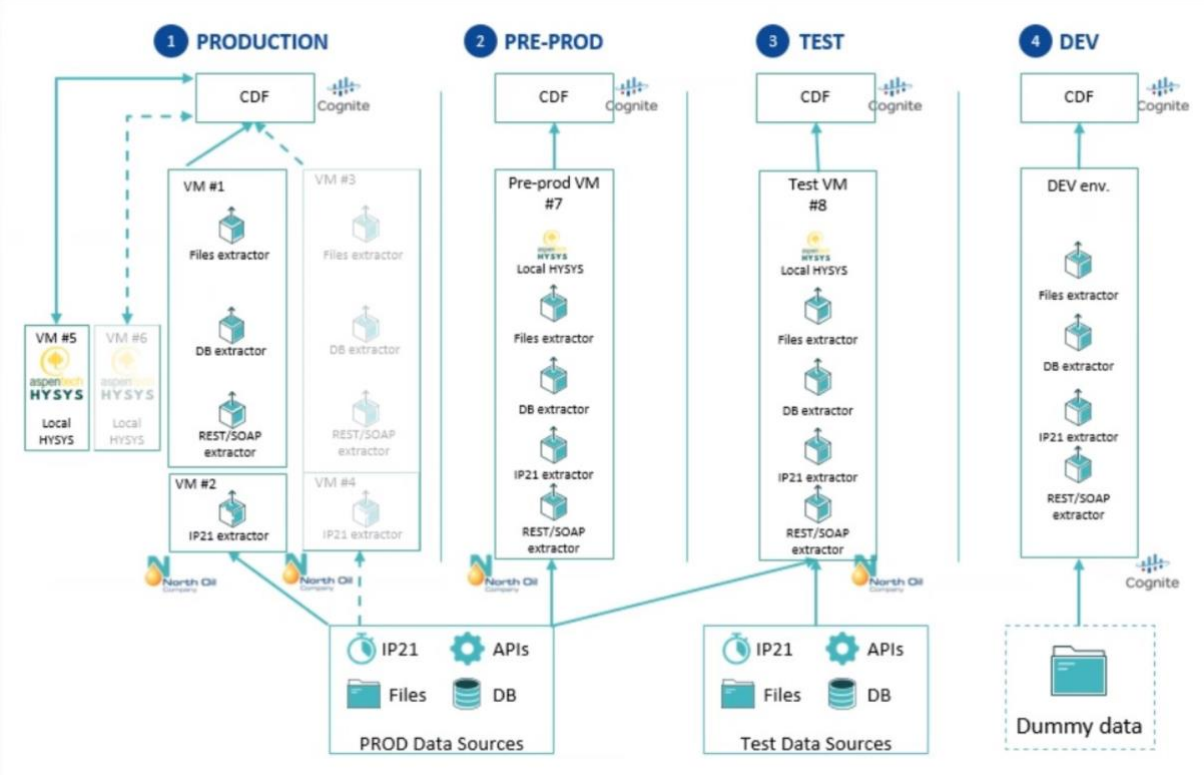
This document will evolve as the core topics are detailed in collaboration between Cognite and NOC.

1 QUALITY

1.1 CORE TOPIC 1 - ENVIRONMENT SEGREGATION

NOC, Metreta, BCG, and Cognite are in full agreement on the importance of having a solid digital architecture and infrastructure as the foundation of the success of the broader digital program and the individual Use-Cases. This will ensure that development teams maintain optimal workflows, in order to achieve high-quality deliverables. A key element of the architecture is the definition and use of multiple environments to ensure that the software is rigorously tested before it is deployed and made available to end-users. Environment segregation supports the robustness and stability of the use-cases and is part of the Quality principle as defined.

MVP#1 Scale-Up architecture will have four independent environments consisting of all relevant components necessary to validate the software. The vertical environments are DEV - TEST - PRE-PROD - PROD representing the project assurance and validation flow. With the horizontal components representing the relevant and necessary components within each layer.



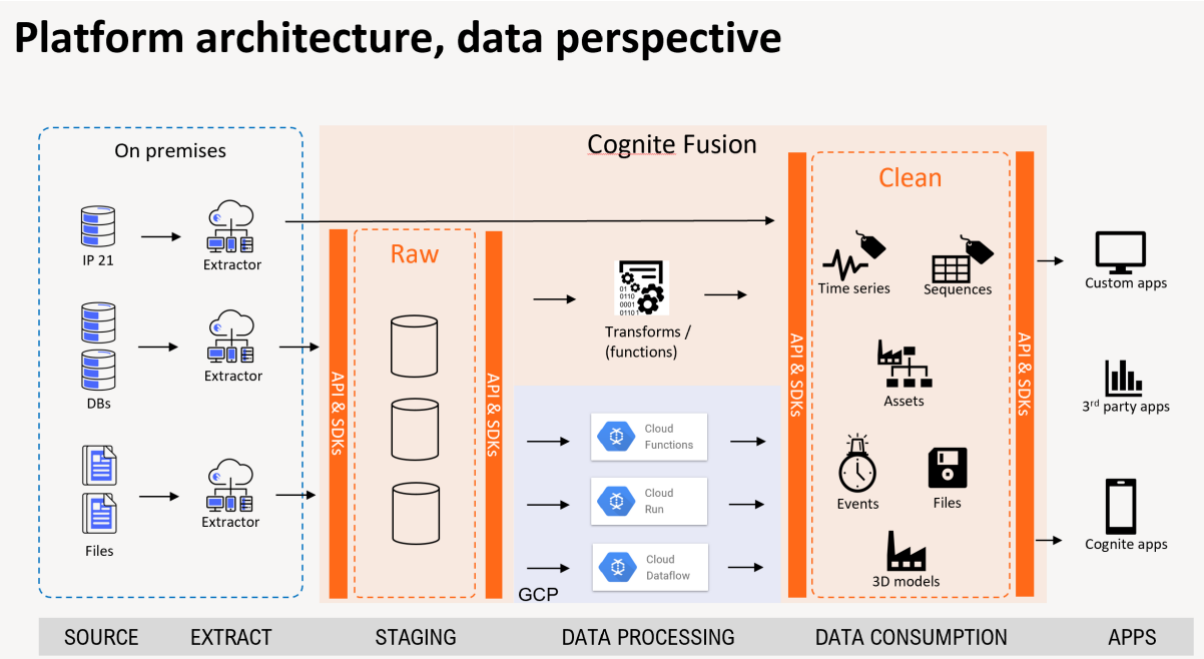
Clarifications	Response
Pre-Prod environment covered by the Test environment: please specify rationale, pros. and cons. of this choice in order to involve NOC in the decisional process;	The suggested setup is a balance between cost (both financial, and complexity) and development bandwidth. Increasing the number of environments: **Pros**: ● Higher test bandwidth (if orchestrated correctly) **Cons**: ● Increased infra costs ● Increased complexity ● Increased risk (due to environments drift)
Specify if tools (e.g., Terraform, etc.) are used to manage the solution infrastructure for different environments;	The Cognite product components (CDF++) are handled by Cognite. We do have a principle of managing our product infrastructure via code, but the actual tooling we use is entirely up to our own discretion. The project components (GCP) are based on "serverless services" so there is minimal infrastructure to manage in the classical sense of the word. Each individual module's deployment is self-contained wrt infrastructure

	specification, and this is managed via GitHub Actions Workflows (per now).
Specify where the ".tfstate" Terraform file is stored (e.g., Locally, remotely, etc.);	N/A
Clarification on PRE-PRODUCTION "Referring to Jorgen's mail (May 18, 11:07 - ""Re: IT Architecture alignment - [14/05] - Architecture Design principles - Sign-off topic 1-Environment segregation 2-DevOps pipelines"": - Referring to the sentence "Cognite is fully responsible for DEV and TEST, and will also help NOC setting up the other environments. NOC is responsible for maintenance of PRE-PROD and PROD environments": it's not clear what are NOC and Cognite responsibilities on PRE-PROD and PROD environments -> Please detail; Referring to the sentence ""Inclusion of PRE-PROD may however increase the risk of negatively affecting use case progression"", we'd like to specify that this task should not impact UCs developments progression, since it's an activity not strictly related to development activities -> Please confirm	We need to separate the flow of data from the artifacts in order to find the best possible set-up for NOC. **Responsibilities** NOC's maintenance of environments • maintain on-premises infrastructure (servers) for that environment • data content of each environment and its completeness (minimal/no deviation) • control access groups membership per environment • deciding on when to promote components on the environment stack • approval of PullRequest promotion between environments • E2E validation and decision making on promoting component between environments Cognite will operate (for all environments) • all corresponding CDF environments and cloud products • Google Cloud Platform project hosting project deliverables • monitor and acting on alerts from project components on each environment • if there is a bug in the project code Cognite is responsible for fixing it • Environment promotion Pull Requests on components per NOC requests Pre-Prod is an often unnecessary middle layer that should be avoided as we normally keep Test (and to a large degree Dev) almost identical to Prod regarding data and artifacts, so that there is not much that can be tested in Pre-Prod that can't be done in Test. Pre-Prod also has a big cost in many aspects, mainly technical debt. Main disadvantages are: • Longer time from development of feature to production, against principle in CI/CD • More complexity in setup and CI/CD pipeline

	• More costs keeping another environment running Cognite will do it's best to not let this cause negative impact on use case progression, and we expec NOC to do the same (wrt the responsibilities set out above)
Clarification on Data replication CDF Data replication mechanisms between PRE-PRODUCTION and TEST must be kept in order to ensure the testability of data coming from data sources not available in TEST (i.e., systems available in PROD but not in TEST) **Further clarifications** The use of the PRE-PRODUCTION environment could help to limit the use of CDF Data Replication mechanisms. Anyway, Data Replication is still needed between PRE-PRODUCTION and TEST, in order to align TEST CDF environment on data coming from data sources that could be not available in TEST. Keep in mind that TEST extractors pull data from TEST data sources and PRODUCTION and PRE-PRODUCTION extractors from PRODUCTION data sources.	This is not recommendable, as mentioned before. We have said earlier that we -could- use data replication to move data between Prod and Test (i.e. it is technically possible). We should however avoid this, and replication should only be done into Dev. We should have separate extractors pushing data into Test and into Prod, the main reasons for this are: ● Replication is not a part of Cognite's products ○ Must be maintained by NOC ○ No updates of replication, contrary to extractors ● Another point of contact ○ Replication introduces another point of contact between source and CDF which inherently increases risk and complexity ● The logic of the replicator is by nature more complex than the extractor itself **Answers to further clarifications** Best practices of DevOps implies that all environments should be completely isolated end-to-end, have similar shape of data but not necessarily the same data values or data completeness. The TEST environment will be treated as Production by Cognite developers and requires a similar shape(schema) of data as the following environments. The only place where replication mechanism should be used is from TEST to DEV environment, as DEV environment is fully controlled by Cognite, should not require external datasource and actively used for development. If the TEST environment doesn't have corresponding TEST data sources available we suggest to use an extractor in TEST infrastructure connected to Production data. That will minimize the initial effort and following maintenance cost in comparison to complex data replication. As mentioned in the clarification above, allowing for replication between pre-prod and test introduces a lot of complexity that might put the delivery at risk. One of them being impossible for NOC to maintain without Cognite

	resources, as replication is not part of Cognite products There are also tasks defined for the non-functional requirements to set up extractors for test, pre-prod and prod. With these, replication will not be needed. For Pre-Prod, we must point to production data sources but the frequency can be much lower in many cases. One also should make sure that the test data sources is a close to real production data as possible in order to test properly.

1.1.1 Detailed Platform Architecture



Source

Source systems are fully governed and managed by NOC. The project depends on data availability and connectivity to the source systems.

Source systems in scope

The below table lists identified and in scope source systems, and what environment they are sending data to. The list is still subject to change, but we do not expect many changes going forward.

Source System	**Description**	**DEV**	**TEST**	**PROD**
SAP	Work orders	N/A	YES	YES
Primavera	Planning	N/A	YES	YES
Synergi	Risks	N/A	YES	YES
eCow	Risks	N/A	YES	YES
D2 (Manassa)	P&IDs and Layouts	N/A	YES	YES
Sciforma	Work orders	N/A	YES	YES
3D models	3D	Manual	Manual	Manual
WellView	Well integrity status and intervention planning	N/A	YES	YES
Fieldbox	Pressure Conditioning and Action Schedule	N/A	YES	YES
Credo	- Remaining life of equipment - Inspection Anomaly Notification	N/A	YES	YES
IP21	Time series for 26 separators and 7 compressors	N/A	YES	YES
POW - EBS	Exception Based Surveillance	N/A	YES	YES
POW - IVM	Results from GAP/Resolve calculations	N/A	YES	YES
POW - VM	Pil, water and gas rates	N/A	YES	YES
PDMS	Detailed liquid constraints	N/A	YES	YES
MOQ Operations	Well master table, Well Test,	N/A	YES	YES

View	Lab Data, Tag mapping			

Clarifications	Response
Please detail the architecture design (e.g., Kubernetes clusters, etc.);	The inner workings of CDF are not exposed to the outside. You interact with CDF via the Cognite api (https://docs.cognite.com/api/v1/). Our underlying modules constantly evolve (and possibly change) so that the services offered through our api are expanded and improved. For the project part of the infrastructure, the modules are listed under 1.1.1.2.2.
Please highlight what is offered as a product and which components are customized;	In general, the orange part of the illustration above (1.1.1.), "Cognite Fusion" is offered as a product, while the blue section (GCP) are project specific modules. However, these modules are still fully managed by Cognite so you will primarily observe the (data processing) services offered by these modules. For extractors (1.1.1.1.) and applications (1.1.1.2.4.) there are explicit definitions listed within those sections

1.1.1.1 On Premise Layer (Extractors)

The extractor is responsible for transferring data from the source system to the staging area. It is primarily a data bridge, not a data processing component (even though the extractor may do some light-weight processing out of convenience or in order to fulfill its primary role).

- Extractor infrastructure is fully managed by NOC (on-premises Windows nodes).
- Extractor deployment and operations is fully managed by NOC (Cognite does not have access to the infrastructure).
- The extractors run as stand-alone components on Windows. Internal state is persisted on the local disk and they push data to CDF via the CDF REST api. Connection management has built-in retry, backoff and restore.
- Extractors are delivered primarily as products to NOC. Please see table below:

Module	Description	Product	Project
IP21	Extracting time series from IP21	Yes	

DB Extractor	Extracting tables from various RDMS	Yes	
Files Extractor	Extracting files from various sources	Yes	
Documentum Extractor	Not currently in use, but may be added pending project requirements. Extracts documents from Documentum D2	Yes	
HYSYS / GAP connector	Workflow/model triggering with input/output wiring	Yes	
Additional	Depending on project requirements. May include HTTP/REST extractor		Yes

Cognite advises that each of the aforementioned modules is deployed on a separate Virtual Machine in NOC infrastructure.

Clarifications	**Response**
Specify and detail eventual technologies / libraries required for the execution of the extractors (e.g., Windows OS for IP21, DLLs, etc.);	Prerequisites for each of the extractors will be presented in their respective release notes. In general: • Windows Server OS, 2016 or newer • IP21 and DB extractors depend on ODBC drivers for connectivity to their respective source systems.

1.1.1.2 Cloud Layers

1.1.1.2.1 Staging (product)

Staging is the default landing area for data extracts. It is an essential "middle man", supporting the data integration process and pipelines. Staging represents multiple roles:

1. Data staging area. Temporary storage for "data handshake" between extractors and transform pipeline, as well as temporary storage for transform pipelines.
2. Data discovery. Interrogating the source data (without touching the source system) in order to help identify how to transform it into the target state.
3. Recording a copy of the source data state for easy re-processing of data. This includes historization.
4. Data storage for derived data sets that are used as input in transform pipelines. Examples include lookup tables with reference data.

Staging is primarily backed by CDF's service "Raw". Raw can be thought of as a wide column store. As all CDF services, Raw is a product delivery. The default interaction with Raw is via the REST api or one of the SDKs wrapping the REST api. The CDF Console also offers a lightweight GUI data explorer for Raw.

1.1.1.2.2 Data processing (project)

The data processing component usually carries the main responsibility for the data model translation. It closes the gap between what the source + extractor delivers and the Cognite data model (CDF.Clean). Main patterns:

- "Raw-to-clean". The default setup for data acquisition integrations where the data is sourced from raw/stage, transformed to the Cognite data model and published to clean. Parts of the contextualization also run in this mode.
- "Clean-to-clean". Enrichment processing of the clean universe where data is continuously uplifted to increase the information value. Also, for data that is extremely latency sensitive, it could be pushed to clean for availability first, and subsequently processed. This is the "eventual consistency" pattern for processing. Parts of contextualization run in this mode.

The data processing logic will be hosted by a combination of Cognite components and cloud infrastructure (Google Cloud). For cloud infrastructure we select modules based on the requirements for the processing. Whenever possible, it is recommended to use fully managed options. The specific requirements for a data processing pipeline will determine which runtime environment we use. In general:

- **Small to medium complexity and small to medium data volume**. These pipelines will use the Cognite Transforms module. Cognite Transforms execute data processing jobs expressed in SQL. This category typically represents the largest share of the data pipelines.
- **Medium to high complexity, small to medium data volume**. These pipelines will use Google Cloud Functions and/or Google Cloud Run, and/or Cognite Functions to host custom code. The code will primarily be Python and leverage the Cognite Python SDK for connectivity to CDF services. This category typically represents the next-largest share of the data pipelines.
- **High complexity, high data volume**. These pipelines will use Google Cloud Dataflow to host custom processing logic. The code will primarily be in Java or Python, using the Cognite Beam Connector and/or the Cognite Python SDK. We expect very few pipelines to need the extreme capabilities offered by this infrastructure.

1.1.1.2.3 Data consumption (product)

Data consumers will normally access data via the services representing the Cognite data model ("CDF.Clean"). More information about the data model: https://docs.cognite.com/cdf/concepts/datamodel.html.

CDF.Clean is a set of services offered by CDF which targets data consumption. The services are accessed via the Cognite REST API or one of the SDKs wrapping the REST API.

1.1.1.2.4 Applications

Application	Description	Product	Project
Cognite Maintenance Planner (UC3)	Interactive 3D (React) application for work order bundling and risk visualisation	YES	
Cognite Digital Twin Cockpit (all)	A place to collect all use cases to support cross functional collaboration - ADI branch - NOT IN SCOPE FOR MVP#1 SCALE-UP	YES	
Grafana (UC2)	3rd party dashboarding tool used in MVP#1 (UC1 & UC2) and potentially used in UC2 Scale Up		YES
Plotly Dash (UC2)	Dashboarding tool based on Python. Potentially used in UC2 Scale Up		YES
Cognite Console	The Cognite Console lets you discover, model, govern, share, and monitor different data from preparation to production, in your cognite data fusion project.	YES	

Clarifications	Response
Specify technologies that underlie each layer in terms of vendor / SaaS service (e.g., Spark Databricks, etc.);	The inner workings of CDF are not exposed to the outside. You interact with CDF via the Cognite api (https://docs.cognite.com/api/v1/) and/or via the Cognite managed GUI, "Console". Our underlying modules constantly evolve (and possibly change) so that the

	services offered through our api are expanded and improved. For the GCP services (still managed by Cognite, so you are not exposed to them), the vendor has descriptions of them: https://cloud.google.com/run, https://cloud.google.com/functions, https://cloud.google.com/dataflow

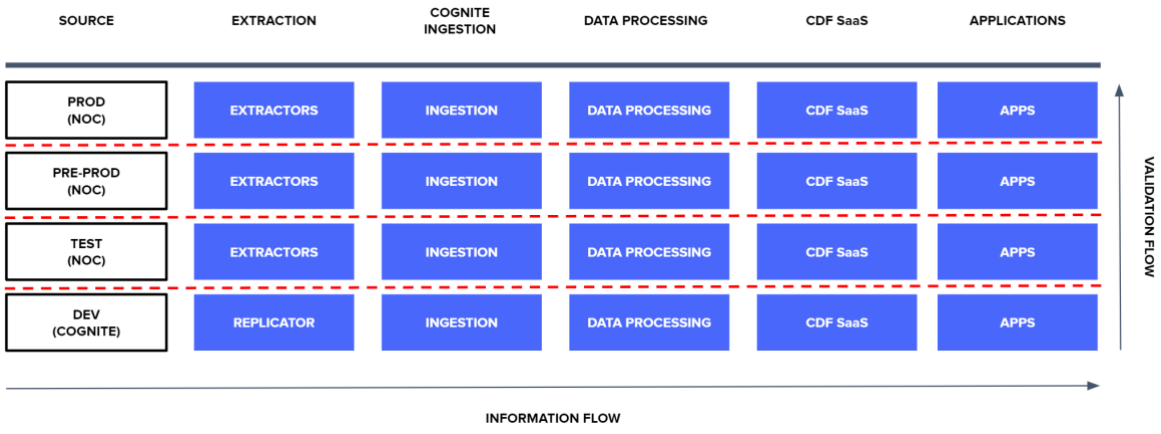
1.1.2 Environments Definition

DT Scale-Up architecture will have four independent environments consisting of all relevant components necessary to validate. Environment segregation supports the robustness and stability of the use-cases and is part of the Quality principle as defined.

The vertical environments are DEV - TEST - PRE-PROD - PROD representing the project assurance and validation flow. With the horizontal components representing the components within each layer.

All environments use the same (type and version of) infrastructure components--albeit fully isolated.

Project Architecture



Dev

- Primary working area for development of code and configuration.
- All code/configuration will go through basic testing in this environment.

- The code/configurations primarily represent data-processing components, so the quality of the tests will depend on how representative (of prod) the source data is.
- Source data should be as representative of prod as feasible. The data characteristics (actual payload) is the most important--the volume of data is less important.
- Access control:
    - Full access for the development teams.
    - End-users would typically not have permanent access to this environment.
    - Temporary access for SMEs helping with validation of data/processing of modules under development.

Test

- Primary area for client acceptance testing of new code/configurations.
    - Ideally, a single change/module is tested at a given time. Must balance risk with test bandwidth.
- Source data should be as representative of prod as possible. Ideally it should have all prod data sets
    - Some historical data may be omitted
    - Some additional data may be present (when testing new integrations)
- Access control:
    - SMEs involved in scenarios under development should have access.
    - Acceptance testing representatives should have access.
    - Key representatives from the development team should have access in order to validate (or debug) modules with prod-data.

Pre-Prod

- Primary area for testing new functionality in an environment that mirrors the production environment as closely as possible.
- Access control:
    - SMEs involved in scenarios under development should have access.

Prod

- Primary area for consumption of governed data.
- Access control:
    - Access for end-users.
    - Access for representatives from the dev teams depends on how closely you want these teams to be able to perform monitoring.

From a development perspective, the ideal scenario is to have prod-representative source data as the baseline for every environment. Thus, for prod, pre-prod and test we have a separate set of extractors feeding data. For dev, we replicate data from the test environment in order to minimise any surprises when promoting the code and to avoid too much load on the source systems.

The main replication points are:

1. CDF.Raw. The Raw service can be replicated on a per database and table basis.
2. CDF.Clean. The various resources in the Clean data model can be replicated based on data set membership.

Synchronization is being done on a case per case basis. It is mainly scheduled to ingest regularly, but we can always choose to sync whenever we need.

Clarifications	**Response**
Specify how the segregation guarantee a full isolation between the different environments (e.g., Dev from Test, etc.);	CDF is set up with a separate tenant per environment, see 1.1.2.2. The GCP services have multiple layers of isolation, with the innermost being "per instance". That is, there is full isolation between each, individual processing module (also within an environment). See 1.1.2.3.
Specify what is referred to Products and what to Custom developments;	See clarifications under 1.1.1.
Detail better the following sentence: "there is no need to keep more than a single (environment-) instance running at a time." ● Does it mean that once you promote an extractor from one environment to the next, the instance running in the previous is stopped? ● Describe rationale, pros. and cons. of this choice in order to involve NOC in the decisional process. **Detail** How is it possible to keep the TEST env. data sources aligned with PROD data sources if TEST extractors are stopped? As stated in the previous row, TEST and PROD will be aligned only through data sources replication (except for a first temporary phase)	Yes, once you promote an extractor from one environment to the next, the instance running in the previous environment may be stopped. Example, promoting the "Synergi extractor" from TEST to PROD. Once the extractor is in PROD, it will feed PROD.STAGE with raw data -> PROD.DataProcessing -> PROD.Clean. This data can then be replicated from PROD.STAGE to TEST.STAGE for running data pipelines in TEST. And/or you can replicate from PROD.Clean to TEST.Clean for running applications in TEST. **Detail answer** Two approaches to populating the (non-prod) environments with data: 1. Feed the environment via a dedicated (and complete) data pipeline. I.e. TEST is populated via extractor.test -> cdf.raw.test -> transform.test -> cdf.clean.test. 2. Feed the environment from PROD data sets. Here, you

	have two replication areas: a. Raw.prod -> raw.test. b. Clean.prod -> clean.test. For "new" data (data that does not yet have a data pipeline in PROD), you have to use option 1. For data that already exists in PROD, you have the possibility of using option 2 if 1) you want to populate TEST with prod-data and 2) you want to populate TEST with data without needing duplicate data pipelines. When using replication, you can set the replication schedule (when to run) as well as replication scope (what data to replicate)
Regarding the Data Replication on CDF layers (CDF.Raw and CDF.Clean) between Prod, Test and Dev, specify that this is a temporary work-around until on-premises VMs and extractors will be set up on the Test environment. As soon as extractors are ready to extract data from Test Data Sources (aligned with PROD data sources), Data Replication mechanisms on CDF layers (CDF.Raw and CDF.Clean) between Prod, Test and Dev will be removed.	You have full ownership over the replication decisions. If you only want to use replication in a temporary manner (i.e. until the TEST extractor VMs are available), then you are of course free to do so.

1.1.2.1 Extractors segregation

- We recommend setting up separate environments (VMs / servers) per relevant environment: dev, test, and prod.
    - Please note the primary dimension of the isolation is per environment--not per extractor instance.
        - You may choose to perform further isolation between extractor instances for the purpose of limiting the blast radius. I.e. If you have some extractor that you judge to be very risk sensitive, you may choose to isolate that instance.
    - Capacity requirements depend on the source data, source system and number of simultaneous extractors running.
    - We expect the prod environment to represent the largest capacity requirements--especially when using replication as the way to sync the different CDF environments.
- Each environment should have the same baseline configuration, i.e. OS and installed software.
- The extractor modules are installed as standalone modules per environment.

- Each environment will have separate connection configurations to target the appropriate CDF environment. I.e. each CDF environment will have a separate API endpoint and identity credentials.
- Each configuration can also target different source systems.

1.1.2.2 Cloud Tenants approach

- The different environments will be served by separate CDF tenants, i.e. data and configuration isolation between the environments. The CDF tenants are:
  - "noc": Production
  - "noc-pre-prod": Pre-Production
  - "noc-test": Test
  - "noc-dev": Development
- The "noc" tenant is hosting today's MVP1 PROD environment. It is recommended to continue to build on this production environment for the scale up phase, using the development and test environments to make a more robust transition. This approach will facilitate keeping the necessary speed to deliver without over-complicating the architecture. Establishment of an entirely new PROD environment for the Scale Up phase is a natural alternative. This will, however, complicate the whole setup and will require double work on "old" and "new" PROD environments for a long time, risking data inconsistency and compromising the project timeline.

1.1.2.3 Cloud Processes and Infrastructure Segregation

- The GCP modules (Functions, Run, Dataflow) run on managed services where each individual instance is isolated from all others. That is the innermost isolation layer.
  - Each function is individually isolated from all the others.
  - Each Cloud Run instance is isolated from all the others.
  - Each Dataflow job execution is isolated from all the others.
- In addition, all GCP modules related to NOC run in a designated GCP project which represents yet another global isolation layer.

1.1.2.4 Cloud Data Segregation

CDF data segregation, inside data owner boundaries, is through logical separation. For example time series data, the majority of the data in scope, is using row key as the isolation identifier. Time series data is currently running on top of managed GCP service Cloud Bigtable.

Clarifications	Response
Detail how data segregation is guaranteed across different data owners (i.e., NOC vs other Cognite clients);	CDF data segregation, inside data owner boundaries, is through logical separation.

1.1.2.5 Application Segregation

Product applications are served as HTML/JS/CSS and run in the user browser, as the user logs in to CDF the application get's data from CDF only from the project that the user is logged in to. Applications might also store non-sensitive data in a specific application backend.

Project specific applications and dashboards are only deployed for NOC and therefore segregation is not relevant.

1.1.2.6 Users Segregation

- Each CDF environment will have separate access control configurations.
- You can re-use user identities across CDF environments by linking the CDF environments to the same user identity provider (Active Directory, etc.). Then the typical end-user will log on using the same identity on all environments, but his/her access level may differ between environments (depending on your configuration).

1.2 CORE TOPIC 2 - DEVOPS PIPELINE

Cognite has had the DevOps mentality embedded in it's culture since the company was founded. High investments in modern infrastructure has been one of the key enablers for being able to build an enterprise ready product in the record time that Cognite delivered CDF onto the market.

For DevOps to not only be a buzzword to mean automation it is imperative that we also follow the rest of the philosophy, meaning promoting accountability, seeking automation, continuous improvement and autonomy. Cognite sees Agile and DevOps as inseparable concepts and will leave a lot of details up to each individual team as these practices need to be ingrained in the individual developer to allow for maximum speed and quality.

This section will not go into detail on internal DevOps practices or tools used to develop or deploy CDF as a platform or other products. It will instead focus on how a complete architecture surrounding CDF and configuration of CDF shall be deployed in the NOC Scale-up project.

1.2.1 Source Code Management

1.2.1.1 Source code repositories

Project code and configuration is developed in a Github project shared between NOC and Cognite. Each team has the autonomy to organize code based on best practices and team preferences. This tends to differ based on type of component and phase of development. This will be documented in each repository as a readme file visible from Github.

In some cases the team might decide to group several components that share dependencies and build processes into a single repository to make development more efficient. This will be documented in the repository.

The release system in Github will be used, all built versions of the software will be available through a UI and APIs from Github.

Access control will be managed by Cognite through Terraform, using code as infrastructure to manage access groups and identities is considered best practice and maintains an easily auditable and automated system for adding and removing access.

Clarifications	Response
Give a preview in this documentation (not only GitHub repositories Readme files) of the main best practices/methodologies that you adopt to organize source code (per team and per component); **Detail** Document the main approaches usually adopted by the different kind of teams / components developed in this document	There is not one size fits all practise to this. There are both different schools of thought as well as individual developer practices. Three examples of approaches you may see from the teams: • Group by dependencies. Multiple components sharing the same dependencies in the same repo. • Group by deployment. Multiple components sharing the same deployment workflow. • Group by role. Multiple components that are representing/covering the same role. I.e. multiple components that together represent one data pipeline. **Detail answer** Please refer to the examples provided earlier. We value autonomy for the teams to define their preferred organization of the repositories and we don't want to put down too hard requirements for them ahead of time.
How the source code of the different layers are organized inside the repository (e.g., extractors, cloud containers, etc.);	Please see above.
Specify what are the different kinds of element stored and versioned inside the repositories (e.g., application source code, configuration files, etc.);	This depends on the component in question, but would typically include: configuration parameters (but not secrets), declarative specifications (i.e. SQL) and code.

1.2.1.2 Branching strategy

Each team is responsible for selecting and implementing a practice for working with version control that suits the team makeup, components lifecycle state and preference. This should be documented in the repositories readme file.

All developers in the project are trained in, and responsible for following good practices for version control.

Clarifications	Response

Please share the information in this documentation, not only GitHub Readme files, going through the suggested ponts: • What is the rationale behind a branch creation and what is its lifecycle (e.g., new feature development, new release, bug fixing, etc.); • How are handled pull requests merges to the Master branch (e.g., requests approval process, reviewers assignment, etc.); • What is the frequency of master updates (e.g., daily merges, etc.). **Detail** Detail the main approaches usually adopted by the different kind of teams / components developed in this document	There is not one size fits all practise to this. There are both different schools of thought as well as individual developer practices. You may see branches based on environment promotion, feature development and/or bug fix. Merges to master (and the environment branches) are typically performed via pull requests and reviews. The reviewer depends on the feature in question + personnel mix. A team may choose to perform daily master merges, but this is not an enforced practice. **Detail answer** Please refer to the examples provided earlier. We value autonomy for the teams to define their preferred branching approach and we don't want to put down too hard requirements for them ahead of time.

1.2.1.3 Secrets / Configurations externalization

The project shall use a combination of Github Actions secret manager and GCP Secret manager to handle all tokens and API keys used in the project. Both of these technologies are fully managed and auditable tools (e.g. taking care of life cycle management, etc). Adopted security measures can be viewed in each of the technologies' documentation.

1.2.1.4 Wiki / Documentation

Each component and repository shall have a Readme file in github that gives an introduction to the component, what configuration is needed and that describes the deployment / test pipelines related to the component.

The Readme will also describe development practices for the repository. The detail level will be of a pragmatic nature, focused on giving the information needed to hand-over or onboard new developers and will be proportional to the complexity of the component.

Documentation about CDF and Products are available through the public website https://docs.cognite.com/

Cognite will also provide a thorough library of documentation about the use cases (problem statement, solution, assumptions and data quality metrics), master tag lists, key functionality, how to get access, and support

Clarifications	Response
Specify why no Wikis (e.g., Confluence, etc.) are used in order to have a well-structured and centralized documentation of all components. Specify also what are, if any, limitations / issues that you could have choosing to adopt them; There are several advantages of using a central dedicated collaboration software like Confluence (e.g., easier to manage centralized places for cross-team documentation, easier to maintain consistency across different kinds of documentations, etc.). Let's understand how we could meet this expectation **Further clarifications** NOC will provide an Azure Wiki instance -> Please confirm that you could use it instead of GitHub Wiki **Further clarifications v2** Our proposal is to use Azure Wiki instead of GitHub Wiki	You can choose to have documentation reflected in the repositories/systems you prefer. In our baseline setup, we don't make any assumptions as to which documentation systems are available--hence the documentation directly in the GitHub repo. If you want to have the documentation copied over to a Wiki (or similar) that you have set up, this is possible **Answers to further clarification** Cognite can copy the content of GitHub wikis to Azure Wiki if that's needed at the end of the project (to make it copy-once). We don't observe high use of GitHub wiki by project teams at the moment, while project documentation exists within source code or README files located in repositories NOC will have full access to. **Answers to clarifications** Currently, use case progress is being documented in Cognie Confluence and readme files on GitHub. Cognite will move documentation to Azure Wiki at the end of the project

1.2.1.5 NOC accessibility

NOC has access to shared Github organization and all documentation stored in repositories. Please refer to section 1.2.1.1.

1.2.2 Continuous Integration / Continuous Delivery

1.2.2.1 Technology stack adopted

Github Actions is the core technology that will be used to create workflows that build and test code based on repository specific requirements. Some repositories will be built and tested with Bazel.

As a default all runners will be based on GitHub-hosted runners (https://help.github.com/en/actions/reference/virtual-environments-for-github-hosted-runners).

The environment segregation with multiple CDF deployments makes it possible to have a low complexity toolchain to support build and development needs in this project.

We also put a lot of freedom and responsibility on the individual development team to choose the exact way to work, this is to allow for state-of-the art tools to be adopted and the most efficient and solid DevOps practices.

Github Actions have live logs and metrics that are available to NOC through the Github UI, there is also a very rich ecosystem of plugins that allow integrations to virtually any major project management system, slack, email etc.

If we at any point during the project should discover that a higher level of orchestration between deployment of components would be necessary we will advance the toolchain to include a more complex deployment service like Spinnaker.

Data Processing components

All **Data Processing code** as described in **1.1.1.2.2** will be built, tested and deployed automatically to the CDF/GCP development environments. Deployment to the test environment will be triggered manually by Cognite after approval of each individual component, this trigger will automatically deploy and schedule code.

The manual approval step can easily be removed and this step fully automated as confidence in stability of code and tests grow as well as a more stable data foundation to run the code on.

Unit testing frameworks are chosen based on the needs and preferences of the developers working on each component. This will be documented in the repositories Readme.

Data Processing in the Test environment will be tested by manually triggered acceptance tests that implement data replication from the production environment, these acceptance tests shall be defined and developed by NOC. As these tests are made available Cognite will set up repositories that contain them and will manage triggering the automatic deployment after NoC has signed off on a release.

Dashboards

3rd party tool with entire configuration and lifecycle management embedded in the tool itself.

Clarifications	Response
Please detail what are limitations, pros. and cons. behind the adoption of specialized CI/CD technologies (e.g., Spinnaker, Jenkins, etc.) instead of the chosen GitHub Actions;	Github Actions is independent of the target infrastructure, flexible and based on a large ecosystem. It is also easy to transition from GitHub actions to another pipeline if necessary. Currently we don't see the need for the extra functionality (and cost) offered by Jenkins and Spinnaker.

1.2.2.2 Pipeline stages details

The pipeline / GitHub actions workflow depends on the module in question. For example, pure configurations (and declarative business logic, like SQL) will not be built, but tested and deployed.

The common workflow stages:

- Build. Builds the core artifacts. For example the container and/or binaries.
- Test. Stand-alone unit-tests and integration tests of the module.
- Deploy. Deploy the artifact to the hosting runtime, for example Cloud Run service.
- Release. Release or promote code to a production environment.

GitHub actions workflows' history can be interrogated on GitHub which provides a complete log of the workflow illustrating the different steps, timings and outputs.

1.2.2.3 Artifacts promotions in the different environments

Build promotions are managed using GitHub. The promotion orchestration is based on tags, branches and pull requests, while the execution of a promotion is handled by GitHub actions workflows.

- The branch represents the current state and deployment history of an environment.
- Promotion to an environment is triggered by a pull request.
    - The promoted artifacts are identified by tags.
    - Promotion is effectively guarded by branch protection.
    - Promotion has an acceptance workflow via pull request approval. For example explicit approval of promotions by NOC.
- The promotion execution (deployment) is handled by GitHub actions.

- The GitHub action workflow run can issue notification per e-mail or via a custom notification action (for example to Slack).

The different development teams adopt peer review practices based on their personnel mix.

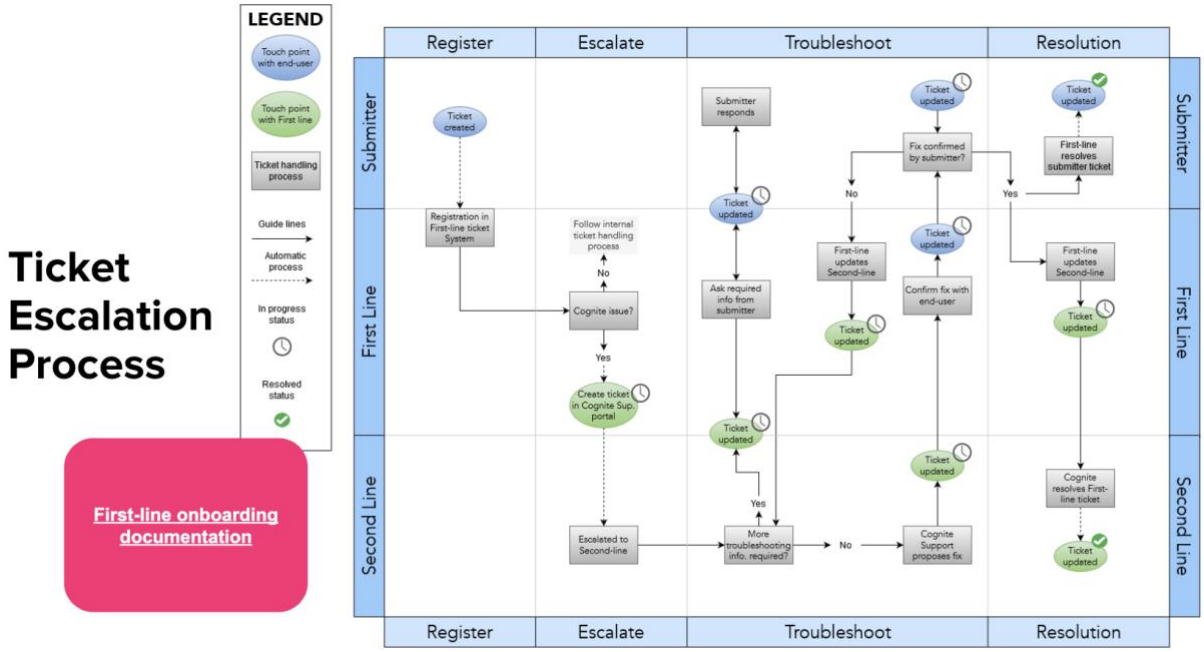
Clarifications	**Response**
Specify how pull request approvals are managed internally (e.g., team leader that approves developers requests, etc.);	The reviewer depends on the feature in question + personnel mix. NOC have indicated that they would prefer to have a sign-off on artifact promotion. Explicit reviewers can be nominated for this.

1.2.3 Issue tracking

1.2.3.1 Processes and tools

Ticket Escalation Process

Ticket escalation processes concerning product delivery have been presented and fine tuned over a series of meetings in April 2020. It is based on the Jira (which contains bug/issue lifecycle) as well as e-mail and MS Teams (for communication purposes).

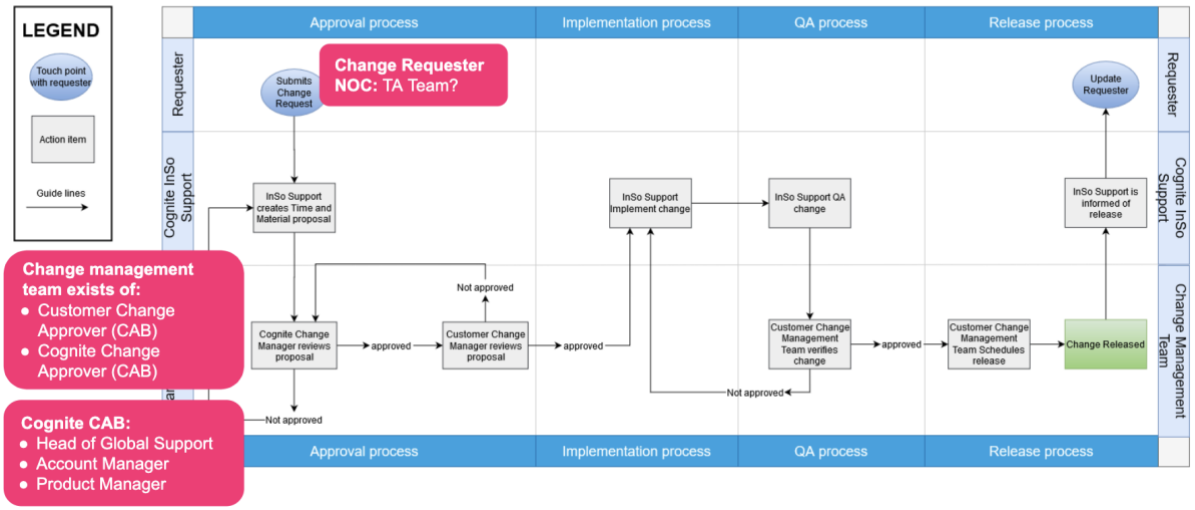


The access to the project related code will be shared through Github repositories. There is a possibility to include links to repositories, pull requests, commits, or build/test steps inside tickets.

Change Management Process

For the feature requests and major changes, the following Change Management process has been presented and agreed upon between Cognite and NOC on 14.04.2020.

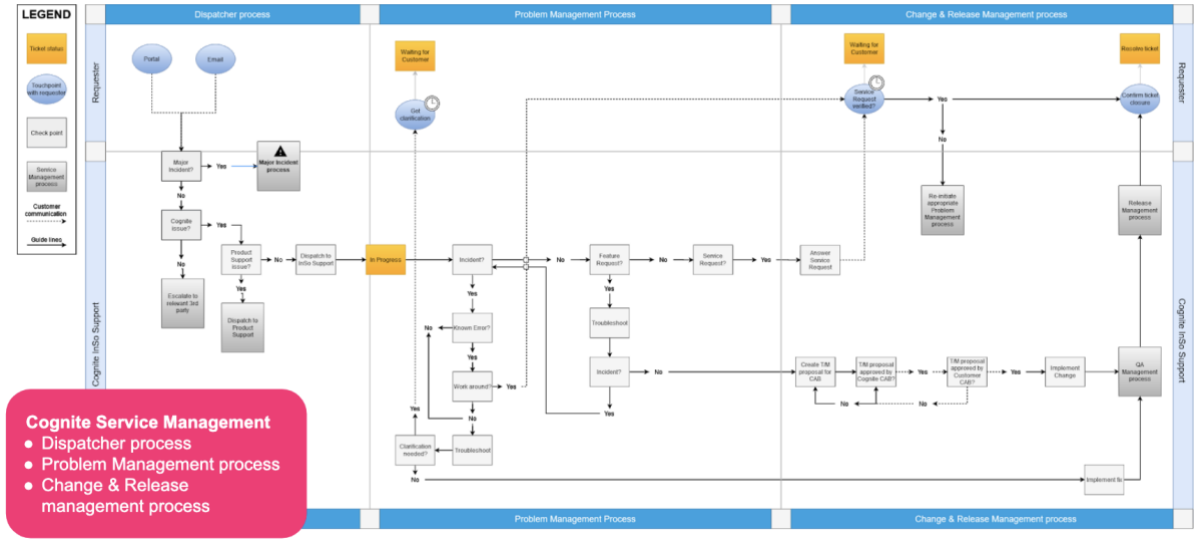
Change Management Process (non SaaS custom solutions)



Incident Management Process

The overall Incident Management Process has been presented and agreed upon between Cognite and NOC on 14.04.2020.

Incident Management Process



Clarifications	Response
Detail how tickets are structured (e.g., ticket template, source code links, severity, etc.); **Detail** Use this documentation to specify also agreed steps	This topic has already been discussed and aligned upon in joint workshop as per April 14th 2020 **Detail answer** The ticketing system is already in place and details about tickets structure can be found here: https://cognitedata.atlassian.net/servicedesk/customer/user/requests?page=1&reporter=org-78 There is a possibility to automatically integrate Jira with Github. If there is a need to help with this topic in the future Cognite will assist. Furthermore, more detailed documentation on this topic has already been shared with the customer and can be linked to the final (after 14 topics) version of this document. Also, although not yet signed, the uptime of supported products will be governed by the standard Cognite SLA (At the time of writing this is is with NOC C&P for review)

1.2.3.2 NOC accessibility

Jira is a selected main issue tracking tool. It contains the entire bug lifecycle process and communication from ticket submission to resolution. The other tools (MS Teams, Trello) have already been put in place during the Scale Up.

1.3 CORE TOPIC 3 - TESTING STRATEGY

1.3.1 Introduction

The testing strategy for the MVP Scale-up project is based on what Cognite considered best practices from modern software engineering. It describes how project teams will work and this mirrors how product teams are working throughout the company. Testing is, in our mind, inseparable from DevOps practices. Therefore a lot of the statements in the previous chapter are relevant to this section.

The goal of this testing strategy is to provide a high quality solution that is possible to deliver within a reasonable timeframe. Good test practices speed up and heighten the quality of software, whereas no or little testing is heavily correlated with low code quality, low speed and high amount of errors in software. However, a misguided focus on reporting has been proven to give the same results as few or no tests.

In this section, we will differentiate between testing of *product* and *project* software and we will only cover testing of product software that is deployed specifically for NOC. It will not cover internal testing practices of CDF features. The goal of the section is to provide the reader with a good understanding about the intention behind testing the different types of components, and the practicalities of how the testing will be performed and how that can be visible to stakeholders.

Clarifications	**Response**
With reference to the sentence "we will only cover testing of product software that is deployed specifically for NOC. It will not cover internal testing practices of CDF features", please provide details about CDF internal testing strategies in order to understand the process behind a new product release in production and possible integration aspects	The current response should give a good understanding of the philosophy and practices also used in Cognite product development. We are not allowed to provide detailed information about the work practices of our product teams to customers, except for in an audit setting Furthermore, as agreed with NOC, Cognite has put in place a process for notifying NOC about all changes in the CDF product portfolio, affecting the MVP#1 or the Scale-Up project. The change management of CDF, APIs, deprecation etc. is governed by the service level agreement. It is also publicly available in our online documentation here on the subject: https://docs.cognite.com/dev/API_versioning.html

1.3.2 Testing practice in Cognite

All developers working for Cognite are expected to write tests for their code. This is a foundational skill in software engineering and we expect developers to not only be proficient in writing tests but to actively seek to use tests to deliver high quality code.

Cognite teaches it's developer to write pragmatic tests, tests shall be written with a value adding purpose in mind and not as a chore.

It is so important to us to attract developers with this mindset that the first paragraph in most of our developer job postings read:

> *"At Cognite you will be encouraged and supported in writing high-quality, testable, maintainable and readable code. We care about craftsmanship and quality and we strive to create applications that impact some of the biggest, most challenging industries today and change the way industries operate tomorrow."*

Testing is considered a core skill needed to be hired in Cognite and we therefore also expect the individual developer to be able to evaluate what tests are required and valuable based on the code, language and context. The developer is also expected to be able to analyze the code, design tests and maintain a testing environment locally and an automated pipeline.

1.3.2.1 Test coverage

Cognite does not enforce any specific test coverage metrics on our code, we believe, as does most modern literature on the subject, that code coverage is not a useful metric to determine quality of code, amount of bugs, stability or value. We encourage NOC or any of it's partners to evaluate the quality of our tests and the coverage of code in a pragmatic fashion. Our goal is and always will be delivery of valuable high quality software.

Following agile software development methodology, our developers evaluate the health of their code base regularly, this includes quality and coverage of tests, complexity and confidence in deployment. As the team is responsible for deploying the code as well as creating it, they are incentivized to write the type of tests that provide value. The developer of any component is the most capable of evaluating where complexity resides in the code and how to write tests that contribute to the quality of delivered software.

1.3.3 Test types

Cognite is responsible for designing and implementing unit and integration tests for project code, NOC is responsible for designing end-to-end, and acceptance tests.
Cognite will provide a framework for running end-to-end tests defined by the customer. NOC is free to run any acceptance tests against products or project code, load, security and pen testing have to be in accordance with contracts and terms of use.

Although not officially signed yet (pending NOC C&P approval), Cognite is responsible for delivering products that conform to our standard SLA (Appendix C to the "Cognite Subscription Agreement" between Cognite and NOC). To achieve this, Cognite relies on vigorous testing of our products as described in **1.3.2**

Clarifications	**Response**
Detail E2E and Acceptance test type, specifying what are Cognite and NOC responsibilities (e.g., design, execution, etc.);	E2E and Acceptance testing process is happening since the beginning of the project: **Phase 1 - Design** Definition of user requirements, expected functionalities and acceptance criteria. Jointly done by the entire project team (Cognite + NOC) with a certain level of fine tuning when we go more into details of the development. **Phase 2 - Execution** Development of the functionalities - currently ongoing. Involves both Cognite development team, NOC PO's, NOC IS SMEs. It is done on the ongoing basis in an agile way driven by the development team. It contains both setup of testing mechanisms and data quality practices where necessary. **Phase 3 - Final validation** Final approval from the NOC end users, handover with final documentation.
Referring to SLAs ("Cognite is responsible for delivering products that conform to our standard SLA""), share the mentioned ""Appendix C to the Cognite Subscription Agreement between Cognite and NOC" **Further clarifications:** Partially covered - SLA document doesn't cover performance SLAs	The mentioned document will be shared as attachment to the next version of this document. **Updated response:** Cognite will attach new version of SLA together with the next version of this document

Security & Penetration Testing: provide here this information in order to have a Testing dedicated section;"	1.3.3.1 updated with security details, moved from CT8.

1.3.3.1 Security and Penetration testing

Security Testing Overview

Cognite use two layers of security (vulnerability and penetration) testing,

1. Internal layer using (automated tools and people)
2. External layer using 3rd party specialist

Internal layer of vulnerability testing:

- Every asset included in scope
- Continuous scanning and validation using in-house as well as 3rd party tools (example include GitHub, Checkmarx Cx-series, Cloud Security Scanner, Container Registry Vulnerability Scanning, and Tenable.io)

External layer (testing completed by external parties/auditors):

- Product every 12 months, tools and method chosen by tester

High-level overview internal layer

The internal layer is the core of the software assurance program where internal security capability and capacity is paired with internal systems understanding and insights to continuously test and validate product and system security. This program is mainly manifested through the Cognite Secure SDLC, starting with the source code and "ending" with continuous monitoring of the running "binary".

From the development process point of view it is important to highlight SDLC components like
Threat Modeling/Architectural Risk Analysis
Peer-Review
Static Analysis
Dependency and License
Automated testing (for example fuzzing)
Manual expert testing (pen-testing)

For internal pen-testing there is a continuous program where components and/or end-to-end services are subject to offensive/red-team activities. The trigger for such activities can

be time since-last-activity or volume/complexity of change. The testing methodology ranges based on target.

Internal layer - Continuous security testing
Through left shifting of security and integrating it into the software development process (SecDevOps) organizations will change how code is tested and releases are managed. When working agile, high release frequency and code velocity, traditional testing/QA cycles will not suffice. Cognite has integrated testing into the DevOps workflow resulting in a methodology often referred to as SecDevOps. What does this mean? It means that security is part of the software development lifecycle and not something that is bolted on afterwards.

Security program related to "software, service, and product assurance"
In the software, service, and product assurance program there are two primary layers

- Internal layer - Operated by Cognite using Cognite personnel and internal/open-source/paid software/services and tools
- External layer using 3rd party expertise/specialists

High-level overview external layer
Cognite will use 3rd party security specialists and auditors to do a complete test, evaluation, and audit of the environment on a 12-month cycle. Below are some highlights of the external layer activities.

External VAPT at least every 12 months
ISO 9001 and 27001
Certification audit completed and passed December 2019
First surveillance audit Q4 2020

SOC 2
Type 1 planned end-of-year 2020
Type 2 planned Q3 2021 (Type 1 + 6 months)

If significant issues or irregularities are detected inside the 12-month cycle it will be adjusted and external party will complete an immediate test, evaluation, and audit of identified issue/irregularity.

Clarifications	Response

Clarify what are security activities mentioned in this section	External Penetration test every 12 months. Result/report will be shared with NOC

Cognite Secure Software Development Lifecycle

Secure SDLC practices is an established methodology and process to ensure a systematic and secure software development lifecycle (SSDL).

Phase 1 - Training

- Security awareness sessions — monthly security awareness sessions on topics like phishing, safe behavior online, password management, and endpoint security.
- Monthly phishing simulations - leveraged to measure effectiveness of and plan awareness training.
- Security training and deep dives — scheduled, upon request by teams, or when introducing a new technology, process, or methodology with impact on security. These include training on threat modeling, identity and access management, secrets management, and others.

Phase 2 - Requirements

- Security requirements — design documents including security requirements are written for new systems and components and are reviewed and discussed in the Architecture Forum meeting place, where the security team is present and assists in identifying and clarifying security requirements.

Phase 3 - Design

- Threat modeling — for specific features/components/systems with security impact, a threat model is done with assistance from the security team when needed. Security team reviews threat models.

Phase 4 - Implementation

- Version Control System (VCS) — all code, including application source code, infrastructure as code, configuration, pipeline definitions, and manifests resides in VCS to ensure change and release management while also providing an auditable change log.
- Code review — all code in VCS is required to be peer reviewed and approved before being merged from a feature branch into a master (production) branch.
- Automated build pipeline — building executable programs, automated quality control gates including functional, integration, and security testing, building of container

images, and deployment are handled by automated build pipelines in a build server triggered by changes pushed to VCS. This allows for consistent and repeatable build steps with quality and security control gates.
- Static Application Security Testing — Security testing with reporting, part of CI/CD pipeline, prior to being approved for production release.
- Software Composition Analysis — Security and licenses scanning with alerts for vulnerable dependencies and use of non-compliant licenses.

Phase 5 - Verification
- Vulnerability Management — Cognite is using several tools to continuously scan for and identify vulnerabilities. Some of the tools currently being used include Google Container Registry Container Analysis, GitHub security alerts, kube-bench, and kube-hunter. Vulnerability management covers activities like network scanning, container scanning, dynamic web application scanning. Detected vulnerabilities are evaluated, ranked, and the mitigation is agreed upon with developers. Vulnerability (issue/bug) is tracked and kept open until mitigated.
- Configuration Management — Infrastructure and configurations are managed as code and this allows for continuous validation of running configuration vs. approved/defined configuration. This protects against platform/environment drift. Changes to platform/environment is subject to test and validation before being rolled out to production.
- Penetration testing — Cognite executes regular security and penetration testing of Cognite Data Fusion services and components.

Phase 6 - Release
- Automated deployments — Automated deployments to Google managed Kubernetes clusters (GKE). Remove the need for developer access to production systems and provide consistent repeatable deployments with rollback capabilities.
- Incident response plan — Cognite does have an incident response plan in place that documents the critical service details, defines playbook requirements, and contact/escalation steps. Services are mapped to- and requirements are defined by the service maturity policy/matrix. The Cognite Service Reliability team is monitoring and ensuring that services are in compliance with the defined requirements.

Phase 7 - Maintenance
- Patch management — Services are redeployed when an update version of the image, service, or dependencies in the container is needed. Google Kubernetes Engine underlying infrastructure is managed and kept up-to-date by Google.

- Logging — Cognite and Google Cloud Platform offers an audit trail of actions when Cognite- and/or Google Cloud personnel interact with your data or related infrastructure. The Audit Trail builds on already robust controls that restrict administrator activity to actions only with valid business justification, such as responding to a specific ticket our customers have initiated or recovering from an outage.
- Security Incident Response — Cognite does have an incident response plan that includes detection and response to incidents. Incidents above agreed threshold and/or impact will be reported through initial report followed by detailed customer postmortem.
- Runtime container security and intrusion detection — Using several tools including GCP native, open source, and commercial tooling/services from Palo Alto Networks.
- Misconfiguration detection and compliance — Using GCP native services, Palo Alto Networks/RedLock, and Forseti Security to ensure continuous scanning and mapping of configurations against standards and best practices.
- Reporting — Monthly security reports are generated and shared with the Cognite organization and senior leadership (including Board of Directors). Where applicable, or requested, relevant information will be shared with external parties (including customers and partners). This will be in addition to the already/contractually committed external reporting cycle.

Clarifications	**Response**
Specify if and how the described solutions will apply to Product vs Custom developments	Project patterns will build on default product patterns and where possible same solution/tool will be used.
Detail the Security Testing process along the whole SDLC (not only as a last step)	Security testing and validation is part of the SDLC/entire product life cycle (from source to running service) and include the following activities: **Phase 1 - Training** • Onboarding security awareness training • Security awareness sessions — monthly security awareness sessions on topics like phishing, safe behavior online, password management, and endpoint security. • Phishing simulations - leveraged to measure effectiveness of and plan awareness training. • Security training and deep dives — scheduled, upon request by teams, or when introducing a new technology, process, or methodology with impact on security. These include training on threat modeling, identity and access management, secrets management, and others.

Phase 2 - Requirements
Security requirements — design documents including security requirements are written for new systems and components and are reviewed and discussed in the Architecture Forum meeting place, where the security team is present and assists in identifying and clarifying security requirements.

Phase 3 - Design

- Threat modeling — for specific features/components/systems with security impact, a threat model is done with assistance from the security team when needed. Security team reviews threat models.
- Architecture Risk Review
- Design Review

Phase 4 - Implementation

- Version Control System (VCS) — all code, including application source code, infrastructure as code, configuration, pipeline definitions, and manifests resides in VCS to ensure change and release management while also providing an auditable change log.
- Code review — all code in VCS is required to be peer reviewed and approved before being merged from a feature branch into a master (production) branch.
    - Enforced by branch protection rules on code repositories.
- Automated build pipeline — building executable programs, automated quality control gates including functional, integration, and security testing, building of container images, and deployment are handled by automated build pipelines in a build server triggered by changes pushed to VCS. This allows for consistent and repeatable build steps with quality and security control gates.
- Static Application Security Testing — Security testing with reporting, part of CI/CD pipeline, prior to being approved for production release. SAST configured to stop the build pipeline or to automatically create Jira (issue tracker) issues. SAST tools invoked via GitHub webhooks or in Jenskinsfile build pipeline definitions:
    - Checkmarx CxSAST
    - Spotbugs
    - PMD
- Software Composition Analysis — Security and licenses scanning with alerts for vulnerable dependencies and use of non-compliant licenses.
    - Checkmarx CxOSA
    - GitHub Security Alerts + Dependabot
- Unit Testing
- Integration Testing

Phase 5 - Verification

	• Vulnerability Management — Cognite is using several tools to continuously scan for and identify vulnerabilities. Some of the tools currently being used include Google Container Registry Container Analysis, GitHub security alerts, kube-bench, and kube-hunter. Vulnerability management covers activities like network scanning, container scanning, dynamic web application scanning. Detected vulnerabilities are evaluated, ranked, and the mitigation is agreed upon with developers. Vulnerability (issue/bug) is tracked and kept open until mitigated.
	
	• Configuration Management — Infrastructure and configurations are managed as code and this allows for continuous validation of running configuration vs. approved/defined configuration. This protects against platform/environment drift. Changes to platform/environment is subject to test and validation before being rolled out to production.
	
	• Penetration testing — Cognite executes regular security and penetration testing of Cognite Data Fusion services and components.
	
	• Dynamic Application Security testing, both as part of regular penetration testing and regular automated tests with Google Cloud Security Command Center Web Security Scanner.
	
	• Image Vulnerability Scanning
	  ○ Container Analysis
	• Fuzzing
	  ○ Custom
	  ○ AFL
	• Network Vulnerability Scanning with Tenable.io
	  ○ Daily scans of publicly exposed interfaces
	
	Phase 6 - Release
	Automated deployments — Automated deployments to Google managed Kubernetes clusters (GKE). Remove the need for developer access to production systems and provide consistent repeatable deployments with rollback capabilities.
	Incident response plan — Cognite does have an incident response plan in place that documents the critical service details, defines playbook requirements, and contact/escalation steps. Services are mapped to- and requirements are defined by the service maturity policy/matrix. The Cognite Service Reliability team is monitoring and ensuring that services are in compliance with the defined requirements.
	
	Phase 7 - Maintenance
	Patch management — Services are redeployed when an update version of the image, service, or dependencies in the container is needed. Google Kubernetes Engine underlying infrastructure is managed and kept up-to-date by Google.
	
	Logging — Cognite and Google Cloud Platform offers an audit trail of actions when Cognite- and/or Google Cloud personnel interact with your data or related infrastructure. The Audit Trail builds on already robust controls that

	restrict administrator activity to actions only with valid business justification, such as responding to a specific ticket our customers have initiated or recovering from an outage. Security Incident Response — Cognite does have an incident response plan that includes detection and response to incidents. Incidents above agreed threshold and/or impact will be reported through initial report followed by detailed customer postmortem. Runtime container security and intrusion detection — Using several tools including GCP native, open source, and commercial tooling/services from Palo Alto Networks. Reporting — Monthly security reports are generated and shared with the Cognite organization and senior leadership (including Board of Directors). Where applicable, or requested, relevant information will be shared with external parties (including customers and partners). This will be in addition to the already/contractually committed external reporting cycle. Misconfiguration detection and compliance — Using GCP Cloud Security Command Center to ensure continuous scanning and mapping of configurations against standards and best practices. ● Continuous asset discovery and inventory ● Security Health Analytics ○ Exposed assets ○ Asset configurations ○ IAM configurations ● Event threat detection (log based) ○ Malware ○ Cryptomining ○ Brute force SSH ○ Outgoing DoS ○ IAM audit ● Container threat detection ○ Suspicious binary ○ Suspicious library ○ Reverse shell ● Security health analytics ○ CIS 1.0 ○ PCI DSS v3.2 ○ NIST 800-53 ○ ISO 27001 ● Web security scanner ○ XSS ○ Flash injection ○ Mixed-content ○ Clear text passwords ○ Usage of insecure JavaScript libraries

What are the source code vulnerability scanning processes adopted for both re-used and new developed code	All code will be scanned as per the SDLC requirements, either as part of the (active development or out-of-band validations including as an example GitHub native scans on a recurring basis (not triggered by developer work).
Detail Penetration testing for each layer (e.g., Application, Infrastructure, etc.)	All components/layers are subject to testing/validation on a high level: ● GCP foundational services/infrastructure - Managed by Google ● Cognite managed infrastructure - Manual and automated testing through the Cognite SAR team. Automated testing/validation by Google on the cloud native technologies and managed tools. ● Application - Manual and automated testing through the Cognite SAR team. External validation by 3rd party on a 12-month cycle. Project software deployed/running on-prem is not included/covered by Cognite manual/automated pentesting. Such is the responsibility of the data-owner/infrastructure operator (in this case NOC).
What are the security tools and solutions (e.g., Burpsuite, Gaunlett, OWASP ZAP, etc.) used to check for vulnerabilities along the development phases (e.g., code, test, deployment, release, etc.)	See replies above/below for tools used. Burp Suite is used as part of the internal testing flow when manual pentest/VAPT exercises are conducted. External testers chose tooling.
How security checks (e.g., tools, solutions, etc.) are integrated into the DevOps pipeline?	Security check and tooling is typically instrumented as GitHub webhook or through configuration/build step files (jenkinsfile). Jenkins-helpers are provided as a library for internal use.
How is security verification done in the Pipeline? They should be detailed as below: ● SAST (white box testing) - software artifacts analyzed to uncover vulnerabilities during the coding and unit testing phases; primarily the source code, and configuration files; ● DAST (black box testing) - a running application is tested for weaknesses; ● Others such as IAST or RASP. ● How are these tests embedded into the pipeline?	Static Application Security Testing is performed by using Checkmarx. Scan is triggered on Pull Requests, feedback with detected issues is included in the Pull Request conversation, is subject to peer review, and Jira tickets are created for issues on merge. Dynamic Application Security Testing is performed using Burp Suite Professional (during VAPT) and Web Security Scanner (built-in service for Google Cloud Security Command Center) on a continuous basis.
The below, not limited to, types of tests should be included and need clarity on: ● Functional security test ○ Targeted at verifying the security features such as authentication and logout work as expected. ● Non-functional tests against known vulnerabilities.	Functional security testing is covered by: ● Penetration testing, internal and external ● Automated tests in Continuous Integration pipeline Non-functional tests against known vulnerabilities: ● Static code analysis (Static Application Security Testing) with CxSAST (Checkmarx SAST) ● Open source dependency scans with GitHub Security Alerts and

○ Targeted tests for weaknesses are known upfront. ○ These should include testing known weaknesses and miss configurations like of HTTP only flag on a session cookie or use of known weak SSL and ciphers etc. ● Security scanning of application and infrastructure. ○ How the content to be scanned is navigated and populated in the scanning tool before starting to scan the application. ● Security testing application logic. ○ Identify flaws in the logic of the application by manual/human intervention. ● Enumerate the attack surface ○ Scan areas which can be attacked, like APIs, command line, open ports, etc. ○ Enumerate the attack vectors test by using different techniques like SQL injection, buffer overflow, XSS. ○ Augment test script so that the proper logs can be obtained	CxOSA (Checkmarx Open Source Analysis) ● Network vulnerability scanning with Tenable.io ● Web Security scans with Google Cloud Security Command Center Web Security Scanner ● Misconfiguration and weak security configuration detected with Google Cloud Security Command Center ● IOC and Threat intelligence with Google Cloud Security Command Center Threat Detection Security scanning of application and infrastructure: ● For infrastructure scanning with Tenable.io, targets are identified from DNS, Google Cloud Platform and Kubernetes APIs. Google CSCC is configured to scan all projects. ● For dynamic application tests (part of internal penetration testing cycles) with Burp Suite Professional, manually navigating helps the application learn about the webapp structure, which the tool's crawler and active scanning capabilities can then take advantage of for better coverage. Security testing of application logic: ● Manually test following OWASP Web Security Testing Guide (internal penetration test cycle) ● Manually test against application's business logic requirements (internal penetration test cycle) Attack surface enumeration: ● Network scans with Tenable.io, with target selection from DNS, Google Cloud Platform and Kubernetes APIs. ● Internally developed security dashboards and alerts (Cognite Security Insights) ingesting data from different sources and systems, both via scanning, audit log ingestion and analysis, and event-driven (webhooks). ● Web application enumeration using manual techniques and tools: Burp Suite, OWASP ZAP, sqlmap, Qualys SSL Labs Server Test.
Detail vulnerabilities reporting	Detected vulnerabilities will have mitigation actions assigned to the applicable teams. Assignments are labeled so that they are tracked and reported on by the Security team. All issues assigned with the 'security finding' label are tracked, in real-time, via internal issue tracking system. These are continuously followed up on to ensure that they have the right prioritization and are brought to completion. Security posture, including vulnerabilities, is reported to management on a regular basis either through recurring meetings or in writing. The cadence of this reporting might vary depending on the amount and severity of vulnerabilities/security findings. Reporting on vulnerabilities is also subject to inclusion in the management review to ensure that any systemic flaws are given sufficient priority and that there is a continuous improvement of the management system.

Detail how vulnerabilities are documented and remediated	Vulnerabilities are documented via issue tracking system (in addition to applicable reports from e.g. VAPT), default at the time of writing is Atlassian Jira. They are further documented through commits, pull-requests and tests in the CI/CD pipeline. In the event of vulnerabilities being related to an incident additional documentation is provided in a postmortem document. Fix forward is the preferred and default method for remediation, roll-back is used as an exemption. Examples of remediations can include: • Code change • Upgrade of dependency • Change to base image • Change to infrastructure configuration

1.3.3.2 Unit and Integration testing

Code that is testable is considered to be significantly more maintainable and solid than code that is not testable, unit testing above anything else is performed to achieve solid code through defining the units specification by writing tests.

Solid unit tests also serve a secondary purpose of documentation of the code unit, making maintenance and onboarding of new developers easier.

Integration testing serves the purpose of proving that a component works in concert with some other components. Regression testing is typically most valuable when written as integration tests, and therefore having a system set up for easily publishing new regression tests for components as integration tests is a requirement for all components.

The exact method of doing unit and integration testing is left to the developers of each component, for all project code this will be clearly documented on the github readme for each component delivered as project code.

1.3.3.3 End to End

End to end (E2E) testing can be done in multiple ways for this project. The purpose of these end to end tests are to serve as the customers acceptance tests for promoting code and configuration to the production environment.

The E2E tests are also set up to continuously run against the production environment, serving the purpose of smoke tests.

NOC has the responsibility of designing these tests and the following three types are set up for implementing that purpose. The tests designed by NOC must be feasible and valuable.

E2E tests can be useful for testing for certain types of regressions, but can become too fragile and hamper future development speed.

End to end - DQM
For time-series data, data quality monitoring will be set up to validate the shape of time-series data at the end of the pipeline. This monitoring of data quality serves as one part of the end to end testing and validation platform.

End to end - UI Automation testing
Tests that automate user interaction with UI to validate user flow based on UC defined patterns that truly validate an end to end case. These tests are typically fragile and are costly to maintain with even small updates to UI. This does mean a high amount of false positives, but when written well a low amount of false negatives.

End to end - Data Tests
Additionally for any needs outside the aforementioned types, we will use Cognite functions or similar FaaS services to perform data tests that could validate the expected shape of data for the different steps of the data pipeline in data sets and in raw storage. This allows us to write expectation tests that would validate data on advanced business defined rules if NOC decides to define such rules.

1.3.4 Testing of various component types

1.3.4.1 Extractors

Extractors are tested internally in Cognite but are also available to test for the customer through black box tests to do acceptance testing. Binaries are provided to the customer through CDF Console UI and the customer is responsible for designing and running the tests. The extractor will have documented behaviour and descriptions on how it would be possible to perform acceptance testing.

Extractors behaviour and, more importantly here, configuration on the NOC project will be covered by end to end tests.

1.3.4.2 Data processing

Python

Data transformations written in Python will have unit tests defined in the same repository, and these tests will run in the developers toolchain when developing locally. All unit tests will run automatically on any new branch in the repository. Details about the testing strategy chosen by the developers will be present in the Readme of the repository. Reports containing coverage of test runs will be available through github actions UI.

Cognite Console transformations - Spark SQL
Data transformations written in Spark SQL to run on Cognite Transformations will have tests defined in the same repository as the code. Tests will run automatically on any new published branch in the repository. Details about the testing strategy chosen by the developers will be present in the Readme of the repository. Reports containing coverage of test runs will be available through github actions UI. The nature of spark SQL queries makes the type of tests written different from procedural languages. Tests are typically written as data tests or integration tests that verify the expected results on a test dataframe.

Clarifications	**Response**
Please follow guidelines shared in the ToC: details the different test phases, covering bullet points topics	The table of contents has been updated to accommodate Cognite's approach. It should still cover all aspects laid out in the ToC provided by NOC
If different developer teams have different methodologies, please, detail them avoiding to refer to future "Readme" files	Our development teams do not have different methodologies but the agile methodology described in 1.3.1 and 1.3.2 is based on autonomy and is what we believe will produce the most valuable end product for our customers.

1.3.5 Testing process phases

The process described in the original ToC does not align with the agile methodology applied to software development and testing in Cognite as described in 1.3.1 and 1.3.2. The developer and team is responsible for all phases of testing for components with the exception of acceptance testing and the definition of E2E tests. The reasoning behind this is thoroughly covered in 1.3.1 and 1.3.2 and we believe that the quality of our work hinges on the accountability and ownership that agile developers working with a DevOps mindset creates.

1.3.6 Test Automation

Clarifications	Response
Specify what are the tools adopted to run / manage test execution	Project: Unit/integration tests are integrated in CI/CD pipelines as actions in Github Actions. For E2E tests see that section. Project testing is typically adopting patterns from SaaS product (see below), this include the following tooling examples (depending for example on the project characteristics): ● JUnit ● pytest ● scalastyle ● scalafmt ● enzyme ● React-testing-library ● testcafe (end to end tests) ● jest snapshot tests ● jest test runners ● supertest for integration tests ● smoke tests ● codecov SaaS product (examples): ● JUnit ● pytest ● scalastyle ● scalafmt ● enzyme ● React-testing-library ● testcafe (end to end tests) ● jest snapshot tests ● jest test runners ● supertest for integration tests ● smoke tests ● codecov
Detail how they are integrated in CI/CD pipelines	Project: Unit/integration tests are integrated in CI/CD pipelines as actions in Github Actions. SaaS product (examples): ● Run on PR and on merge into mainline, example instrumented through ○ jenkinsfile ○ package.json
Detail how these tools are integrated with other Cognite tools (e.g., Jira, etc.) and how could be integrated with NOC relevant tools (e.g., Azure DevOps, etc.)	Project: Unit/Integration tests can be integrated with almost any system through Github Actions see a list of possible connections in their publicly available documentation here:

	https://github.com/marketplace?type=actions Product: Integrations and tool selection is team specific as an overview Cognite is using default Jira but teams can also use GitHub Issues, Zenhub, and Trello. Integration between Cognite SaaS Product and NOC is not in scope, such pattern is not supported.

1.4 CORE TOPIC 4 - DATA QUALITY / GOVERNANCE

1.4.1 Data quality

1.4.1.1 Data quality assurance

Data quality checks can be set up in the data processing modules on a per case scenario and send alerts.

The data quality is focused on pragmatic data quality as defined by the ISO 8000 standard, which is aligned with NOC's needs. This means that the degree to which data is appropriate and useful is defined on a per use case basis. This is a more strict requirement than syntactic or semantic quality.

Clarifications	Response
Exactly what data quality checks are available in the data processing modules? How is it possible to create and manage quality checks directly by NOC?"	Please navigate to our publicly available documentation to read about CDF features like this. https://docs.cognite.com/cdf/data_governance/concepts/data_quality_monitoring/ Or visit our product on https://console.cognitedata.com and navigate to data quality monitoring.
Further clarifications Detail Data Quality checks on interconnected data (e.g. information mismatch/clash from 2 different data sources, etc.)	**Answers to further clarifications** Data quality evaluations are guarded by business rules and informed by subject matter expertise. When doing checks on interconnected data, we would typically interrogate subject matter expertise as a part of the scenario workstream. For example, system owners and data stewards representing the sources in question as well as scenario experts representing the target user workflow/application. Based on this input, we would implement a set of targeted business rules (as a part of the data processing pipeline) to perform the checks.
Further clarifications v2 Regarding the clarification about "Data Quality checks on interconnected data" specify what "specific subject matter expertise" do you refer to for each data typology (e.g., owner of the data source, data quality subject matter expert, etc.).	**Answers to further clarifications v2** We suggest that the client defines a "Data Quality Owner" role, which is someone who understands the source systems and has an overview of how they relate. This person is responsible for comparing the output from CDF with the actual content in the data sources. He/she does not need to have a deep understanding of the different data sources, but should be able to know who to contact if in-depth knowledge is needed.
Further clarifications v3	**Answers to further clarifications v3**

• NOC SMEs will provide required business rules; • Cognite will take care of implementing specific business rules in case of the provided console have some limitations • In general Cognite should provide automation to implement data quality checks. Where automation could be not implemented, Cognite should provide a functionality to manually perform checks (e.g., data integrity checks, etc.) **Further clarifications v4** Specify responsibilities behind manual checks implementations (i.e., Cognite and NOC responsibilities), using the provided SDKs	Without knowing the type or scope of these business rules Cognite cannot commit to implement these. Depending on the scope of these business rules, the amount of work should be estimated and discussed. It is possible for Cognite to implement these as part of the CICD pipeline. Our Python SDK, or JS SDK is a simple way of doing manual checks (e.g., data integrity checks) **Answer to further clarifications v4** The Cognite SDK gives full flexibility in using the data in CDF. Anyone with a service account can use this to extract the data needed for analysis (as long as the user has read access to that data) and perform any type of check he or she wants. This is the preferred way of doing ad hoc analysis. Manuels checks done during development by Cognite is Cognite's responsibility. Any manual checks by NOC is NOC's responsibility.

1.4.1.2 Data cleansing

Different types of data cleansing happens at different stages, depending on the need of the use case. The tooling used to clean data is mainly Cognite Functions or CloudRun, used by the use case crews.

Clarifications	**Response**
Please detail the data cleansing steps more in detail. **Further clarifications** Cognite will provide a Report or UI to compare data quality before and after data cleansing **Further clarifications v2**	This is described by the code that does the cleansing (python code, SQL transformations) and is defined case by case. Answering this generically does not bring value, and thus it will be available in GitHub as part of the documentation. **Answers to further clarifications** Data quality monitoring in Cognite Console can be used to monitor data points in time-series both before and after a this has been changed regardless of purpose. For other resource types there are no built in tools to report on data quality in a data pipeline. There is planned development on such a feature over the next 12-24 months for CDF. Please specify if you see specific needs that we should make sure to cover. **Answers to further clarifications v2**

• Detail the main patterns used for data cleansing, including and not limited to:
  ○ Data Profiling: any form of data analysis used to inspect data and assess quality.
  ○ Data Enhancement/Enrichment: any process used to add attributes to a dataset to increase its quality and usability.
  ○ Data Parsing and Formatting: any data analysis process using pre-determined rules to define its content or value.
  ○ Data Standardization: any data transformation used to make data conform to a standard or a domain rule.
• Detail how NOC could be able to compare data quality before and after data cleansing (e.g., Report, UI, etc.)
  ○ comparing the overall quality metrics & KPIs before and after data cleansing.
  ○ comparing object/values before & after data cleansing. For example value 'X' became 'Y' or object 'A' merged with object 'B',..etc as part of the cleansing so we need a kind of UI/report for that.
	We use the data quality monitoring in the Console to check the quality of time series data. Other quality assessments that are not covered by this must be considered case by case, and can be developed using out SDKs. Data enhancement and enrichment is covered by Transformations or custom code
	
	If one wants to compare data before and after cleansing, one has to make this report oneself.

Further clarifications v3 • Specify responsibilities behind other quality assessments implementations (i.e., Cognite and NOC responsibilities), using the provided SDKs; • Specify if the CDF capability to compare data quality before and after data cleansing is in the product roadmap	**Answers to further clarifications v3** • NOC is free to make any ad hoc test they like and use our SDK as the main tool to access and read data from CDF. • One can make a report and extract data before data cleansing and then extract data after the cleansing and compare the results. Automatic reporting on this is not on the roadmap.
Further clarifications v4 • It's an acceptable workaround for some cases (like value changed from 'X' to 'Y') but in some other cleansing scenario (Object A merged with Object B for example) this will not work. Please provide a list of possible scenarios of the cleansing outcome and workaround for each	**Answers to further clarifications v4** There should be an NOC ownership to both the source system data model and quality, and consumer-side (i.e. data as exposed by CDF.clean) data models. That is, the UCs together with NOC system owners need to be a part of validating and approving the end result from a data pipeline. Included in this is any data cleansing and enrichment logic. Data can always be compared "before" vs. "after" also in the cases where we merge multiple source data collections together to a single target collection.

1.4.1.3 Data quality control

Cognite Console has a user interface in which you can visualize problems with configured **data quality monitoring**. A monitoring dashboard can contain all the time series data that

is being used in a data model to monitor the health of the system. We use rule sets to specify data quality requirements, and continuously ensure that the data meet the requirements. Each data object in CDF can belong to multiple monitors.

A framework for deploying NOC specific quality checks in place should be based on the use case needs.

1.4.2 Data governance

1.4.2.1 Data storage transparency

A high level overview of the cognite data model can be found at https://docs.cognite.com/cdf/ It is also explained in chapter 1.1.1.2.

Technologies and SaaS services being used as a persistence layer is part of the DF product, and hence subject to change. Customers interact with ignite's APIs, and the underlying modules are abstracted away. Over time, these technologies and services change. As such it would not make sense to disclose the current setup, as it quickly will become obsolete

Each object in CDF has timestamps that specifies when it was created and last updated.

Data lineage is documented by **data sets** and internal audit logging in the product. Data sets let you document and track data lineage, ensure data integrity, and allow 3rd parties to write their insights securely back to your Cognite Data Fusion (CDF) project. Data Sets group and track data by its source. For example, a data set can contain all work orders originating from SAP, or the output data from a 3rd party partner's machine learning model. Typically, organizations have one data set for each of its data ingestion pipelines in CDF

A data set is a container for data objects and has metadata with information about the data it contains. For example, you can use the data set metadata to document who is responsible for the data, upload documentation files, describe the data lineage, and so on. Data sets are represented in CDF as a separate resource type with a "/datasets" API endpoint.

As data sets are first class citizens in CDF, with full access via the Cognite API, you can integrate it with a data catalog of your choice.

To define which **data objects**, for example events, files, and time series, belong to a data set, you specify the relevant "dataSetId" field for each data object. This is typically done

programmatically in the data ingestion pipelines. Data objects can belong to only one data set so that you can unambiguously trace the data lineage for each data object.

Clarifications	**Response**
Detail how using Cognite provided DataSet it is possible to effectively implement a full data lineage functionality which gives information, for each data entry, about: • which data source(s) generated it • which transformations happened on the raw data (e.g., which component applied them, which version of the component was involved, what transformation has been applied, etc.)"	This is described in the paragraphs about data sets above. Please also check the data set feature in Cognite Console and https://docs.cognite.com/cdf/data_governance/concepts/data sets/

1.4.2.2 Data flow transparency

Console Transformations module deals with medium complexity and data volume. With Transformations, you can use Spark SQL queries to transform data from the CDF staging area, RAW, into the CDF data model. CDF Transformations is an integrated part of CDF, and you can run it in your browser.

More complex transformations are done in code using the api. See also chapter 1.1.1.2

1.4.2.3 Data freshness

See data quality monitoring in chapter 1.4.1.3

2 SCALABILITY

2.1 CORE TOPIC 5 - IT SOURCE CONNECTIVITY

2.1.1 Data extractors and data sources

Module	Description	Product	Project
IP21	Extracting time series from IP21. Runs continuously (i.e. "streams data") as a Win service. Configured via a local config file.	Yes	
DB Extractor	Extracting tables from various RDMS. Runs on a schedule (batch) with run configuration specified in a local config file. More docs: https://docs.cognite.com/cdf/integration/guides/extraction/db.html	Yes	
Files Extractor	Extracting files from various sources. Runs on a schedule, with run configuration specified in a local config file.	Yes	
Documentum Extractor	Not currently in use, but may be added pending project requirements. Extracts documents from Documentum D2. Runs on a schedule, with run configuration specified in a local config file.	Yes	
HYSYS / GAP connector	Workflow/model triggering with input/output wiring. Runs on a schedule, with run configuration specified in a local config file.	Yes	
PI extractor	Streams data from PI. More doc: https://docs.cognite.com/cdf/integration/guide	Yes	

	s/extraction/pi.html		
Additional	Depending on project requirements. May include HTTP/REST extractor		Yes

For plans regarding the development of new typologies of data extractors and protocols (e.g., REST APIs extractors, SOAP-based extractors, etc.), we suggest hosting a roadmap session with Cognite product management

For details on technology used within the data extractors (e.g., specific 3rd party library dependencies, containerized solutions available for each extractor, OS limitations, etc.), please refer to clarifications in section 1.1.1.1.

For defined data sources, connectivity is recommended to approach on a use case basis. Each Use Case will investigate every source system for connectivity options as well as prepare a solution proposal for NOC to approve

Clarifications	**Response**
Details about technologies behind data extractors: • Specify what are the already available Release Notes mentioned and how they are accessible (e.g., shared folder, etc.); • Specify when remaining Release notes will be available	There are extractor documentation/release notes available for the db-extractor + PI and D2. These can be shared as files via a shared folder. The remaining extractor documentation (IP21, File, Hysys/Gap) will be made available by the end of May.
Documentation on extractors configurability: provide documentation about extractors configurability on cross use cases aspects. As agreed, configurations specific for UCs will be detailed in UC documentation	The generic extractor configuration documentation is a part of the release notes / extractor product documentation mentioned above.

2.1.1.1 Access to data sources

Extractors primarily perform data transport, not processing/transformation. How they interact with the source system depends on the source system itself. In general an extractor will:
Query the source for a collection of objects.
Receive the data in batches.
Commit each batch to CDF and then move to the next batch.

Push or pull approach depends entirely on the source. In general, if the source has push-capabilities, there's rarely a need for an extractor. Then the source would just push data to the CDF REST API. The most common scenario is that the source does not have push-capabilities, so we need an extractor to pull data from it.

Failure recovery providing zero data loss
In batch extract scenarios (the most common), the extractor would re-run the previous data extract specification at the next scheduled run.

For streaming, the extractor would perform re-tries at scheduled intervals in order to try to recover either source or target downtime. If the extractor itself has downtime, it would recover the lost time periods by comparing the source data time ranges with the CDF data time ranges.

The zero data loss requirement is fulfilled by all the different data extractors with the same approach

The IP21, PI, Database and D2 extractors have built-in capability to target new or changed records in order to provide a delta extraction and not a full data dump. Note, most of the extractors depend on a stable protocol and data format from the source. If the source breaches that protocol it will be logged + notified in a metric, and the data transfer will not progress.

Some extractors, like the file extractor, have an inherent flexibility. For example, a file format can change without the extractor reacting to it (since it just transfers the binary).

All extractors allow minimal impact on the data source to avoid contention with critical source processes. Some use built-in throttling, some use delta-load specifications to minimize the source impact.

2.1.1.2 Data Extractors Installation & Configuration

The extractors are configured via a local config file. The config file across extractors follow a similar layout. In addition, comprehensive documentation about extractors configuration is provided together with the extractor itself, tailored per extractor.

Extractors are provided as binary artifacts. Customers may choose the deployment method based on this. Most customers deploy them manually on their host systems.

2.1.2 Write-back

Extractors does not provide write-back functionality for any of the available data sources in NOC. The extractors read data from the data source and write the data to CDF.

To provide such functionality, two main approaches should be considered:

- The source system (i.e. Primavera, SAP) acts as a data consumer and pulls data from CDF. In this scenario, the data to be fed back to Primavera/Sap would be inserted to a specific dataset in CDF which Primavera/Sap reads from. This pattern is used to close the maintenance loop "notification style".
- A specific write-back agent writes directly to the Primavera/Sap API. This approach offers higher fidelity, but it also requires more effort to implement.

2.1.3 Workflow management

For planned Input and Output interfaces of the platform for integrating in the workflows of NOC, All Cognite interfaces are available at https://docs.cognite.com/api/v1/, and https://docs.cognite.com/api/playground/. The common pattern is that the client/consumer needs to poll CDF for updates. I.e. "has this data object arrived?", "has this job completed?". Typically, workflow logic does not reside inside CDF, but in the use case logic/applications.

If parts of the CDF services are unavailable, the API will respond according to the HTTP error codes. If non-CDF components (i.e. components providing data to CDF) are unavailable, then these components need to communicate their status--or a separate monitoring component needs to do so. In other words, each agent/module in the data flow chain needs failure state awareness.

2.2 CORE TOPIC 6 - HIGH AVAILABILITY AND DISASTER RECOVERY

2.2.1 High Availability

2.2.1.1 Cloud platform High Availability

Cloud provider

The provided information is a reference for Google Cloud Platform (GCP), retrieved from GCP public documentation (i.e. https://cloud.google.com, https://cloud.google.com/products, and https://cloud.google.com/terms/sla). Further details and updates can be retrieved from the GCP SLA site.

Furthermore, the list below reflects a summary of services. Note however that it is non-exhaustive. The accuracy of the provided information/details is based on this and further dependent on the delta between 2020-MAY-12 and the current date. Where necessary additional details/information should be part of the use-case documentation.

Monthly uptime percentage

- Compute and storage
  - Compute Engine and GKE https://cloud.google.com/compute/sla
    - Multi-zone instances >=99.99%
    - Single instance >=99.5%
    - Load balancing (as part of Compute Engine service) >=99.99%*
  - Cloud Storage
    - Standard storage class in a multi-region or dual-region location of Cloud Storage >=99.95%
    - Standard storage class in a regional location of Cloud Storage; Nearline or Coldline storage class in a multi-region or dual-region location of Cloud Storage >= 99.9%
    - Nearline or Coldline storage class in a regional location of Cloud Storage; Durable Reduced Availability storage class in any location of Cloud Storage >= 99.0%
  - Cloud Bigtable https://cloud.google.com/bigtable/sla
    - Cloud Bigtable - Replicated Instance (2 or more clusters)
      - with Multi-Cluster routing policy >=99.99%
      - with Single-Cluster routing policy >= 99.9%
    - Cloud Bigtable - Zonal instance (single cluster) >= 99.9%

    - Cloud Spanner https://cloud.google.com/spanner
        - Cloud Spanner - Multi-Regional Instance >= 99.999%
        - Cloud Spanner - Regional Instance >= 99.99%
    - Cloud SQL https://cloud.google.com/sql/sla
        - MySQL and PostgreSQL will provide a Monthly Uptime Percentage to Customer of at least 99.95%
- Networking and data transfer
    - Cloud Load Balancing:
        - * See Compute and storage
    - Cloud DNS:
        - Serving DNS queries from at least one of the Google managed Authoritative Name Servers =100%

Service features

- Compute and storage
    - Compute Engine:
        - Scalable compute resources
        - Predefined and custom machine types
        - Fast boot times
        - Snapshots
        - Instance templates
        - Managed instance groups
        - Reserved Instances
        - Persistent disks
        - Live migration
    - Cloud Storage:
        - Highly durable object store
        - Geo-redundant storage
        - Storage classes
        - Object lifecycle management
        - Data transfer from other sources
        - Encryption at rest by default
    - GKE:
        - Managed environment for deploying and scaling containerized applications
        - Node auto-repair
        - Liveness and readiness probes
        - Persistent volumes
        - Multi-zone and regional clusters

        - Command-line tool for managing cross-regional clusters
- Networking and data transfer
    - Cloud Load Balancing:
        - Health checks
        - Single Anycast IP
        - Cross-region
        - Cloud CDN integration
        - Autoscaling integration
    - Traffic Director:
        - Google-managed Global L7 ILB
        - Control plane for xDSv2-compliant open service proxies
        - Supports VMs and Containers
        - Health check offloading
        - Rapid autoscaling
        - Advanced request routing and rich traffic-control policies
    - Cloud DNS:
        - Programmatic DNS management
        - Access control
        - Anycast to serve zones
    - Cloud Interconnect
        - Cloud VPN (IPsec VPN)
        - Direct peering
- Management and monitoring
    - Cloud Status Dashboard:
        - Status of Google Cloud services
    - Google Cloud's operations suite
        - Uptime monitoring
        - Alerts
        - Logging
        - Error reporting
    - Deployment Manager
        - Repeatable and consistent deployment process
        - Parallel deployment
        - Templates
        - Infrastructure as code

Cognite

Cognite has multiple levels of redundancy in Cognite Data Fusion and also utilizes relevant backup, snapshot and data redundancy mechanisms in the Google Cloud Platform IaaS.

Resource type	Data type	Storage service	Backup frequency	Retention
Timeseries*	data points	Bigtable	Weekly	4xweekly+2xmonthly
Timeseries	metadata	CloudSQL	daily	7 days
Sequences	data points	Bigtable	Weekly	4xweekly+2xmonthly
Sequences	metadata	CloudSQL	daily	7 days
Assets	(all)	CloudSQL	daily	7 days
Events	(all)	Spanner	daily	28 days
Files	"the file itself"	Google cloud storage	11 9's	28 days
Files	metadata	CloudSQL	daily	7 days
3D	metadata	CloudSQL	daily	7 days
3D	model data	Google cloud storage	11 9's	28 days
Console	user settings	Firebase	daily	28 days
InField Application	user settings	Firebase	daily	28 days
Asset Data Insight Application	user settings	Firebase	daily	28 days
Auth	groups, service accounts, API-keys	CloudSQL	daily	7 days
Auth	Group memberships	CloudSQL	daily	7 days
Audit Logs (hot)	logs	BigQuery	continuous	7 days
Audit Logs (cold)	logs	GCS	daily	6 weeks

*Snapshots are stored in Big Table:

- Snapshot takes 5 minutes
- Write of Snapshot to different table 5 mins (creates copy of initial snapshot)
- Write of snapshot to GCS 5 days (the copied snapshot is backed up)

Google Cloud Storage

- Used for file content storage. Dependent on built-in durability guarantee.
    - 99.999999999% (11 9's) annual durability

- https://cloud.google.com/storage/docs/faq#policy

Kubernetes HA principles/configurations

As a principle Cognite is using native HA/resilience where possible, hence avoiding writing/maintaining own logic.

Cluster Autoscaler
GKE's Cluster autoscaler automatically resizes the number of nodes in a given node pool, based on the workload demands. You don't need to manually add or remove nodes or over-provision your node pools.

Cluster autoscaler works on a per-node pool basis. When you configure a node pool with cluster autoscaler, you specify a minimum and maximum size for the node pool.

Cluster autoscaler increases or decreases the size of the node pool automatically, based on the resource requests (rather than actual resource utilization) of Pods running on that node pool's nodes. It periodically checks the status of Pods and nodes, and takes action:

If Pods are unschedulable because there are not enough nodes in the node pool, cluster autoscaler adds nodes, up to the maximum size of the node pool.
If nodes are under-utilized, and all Pods could be scheduled even with fewer nodes in the node pool, Cluster autoscaler removes nodes, down to the minimum size of the node pool. If the node cannot be drained gracefully after a timeout period (currently 10 minutes), the node is forcibly terminated. The grace period is not configurable for GKE clusters.
If your Pods have requested too few resources (or haven't changed the defaults, which might be insufficient) and your nodes are experiencing shortages, cluster autoscaler does not correct the situation. You can help ensure cluster autoscaler works as accurately as possible by making explicit resource requests for all of your workloads.

Using native Kubernetes functionality - Deployments:

- Each type of Kubernetes pod is managed by a Deployment, and a Deployment can manage multiple replicas of the same pod. Deployments use ReplicaSets to provide self-healing and scalability:
  - If a pod fails, it will be replaced - self-healing
  - If load on a pod increases, this is handled by adding additional pods - scaling
- Deployments also enable zero-downtime rolling updates. The Deployment YAML file is updated with a new image version and reposted to the (Kubernetes) API server,

the new desired state is registered on the cluster requesting pods to run the new version of the image.

Dashboard HA principles/configurations

Principle(s)

- Dashboards hosted/run on the default Cognite containerized HA infrastructure

Application Content Delivery

This section will also be referenced in upcoming Core Topic 7 as it touches on performance and availability.

Please reference the Cognite Tech Blog article on supercharging our web applications.

2.2.1.2 On-premises components High Availability

An assumption of on-premises components' hosting environment is configured for high availability (HA) must be made.

Extractors can be configured for HA by separating the three types (DB Extractor, Files Extractor and Documentum Extractor) in mutually redundant environments and choose one of the following options:

- Each extractor is configured with an active instance in it's main environment and a passive instance in one of the others in a manner that allows for failover if one of the environments should become unavailable.
- Each extractor is configured with an active instance in it's main environment and a passive instance in both of the others in a manner that allows for failover if two of the environments should become unavailable.

Clarifications	Response
Referring to On-premises components, describe a suggested approach to automatically detect extractors unavailability (e.g., CDF feature that analyze extractors data, on-premises feature to install inside VMs, etc.) and trigger actions (e.g., automatically re-start VMs, etc.) or alerts (e.g., send emails, etc.)	Monitoring and orchestration of extractor/VM HA behaviour is recommended managed by NOC using existing infrastructure observability and technical support stack including consumption of extractor telemetry/insights. Where necessary it is recommended that extractor supporting infrastructure is established, this can include Prometheus and Grafana for metrics collection, alerting, and visualization.

	Cognite Console data quality rules and alerting can be used as a trigger based on loss of data or reduced quality (rule based trigger). Trigger can be mail to NOC and NOC can build workflow originating from such triggers. Configuration and tuning of such monitoring should be part of the use-case workflow in order to ensure accuracy and relevance.
For DR and HA purposes detail how the shared state of different instances of the same extractors is handled by the system (e.g., input given by CDF directly, etc.) in order to resume the extractor execution from the last available extracted data **Further clarifications** The scenario proposed regards an active-active scenario only. The purpose of the question was to understand how a passive instance of a streaming data extractor can recover from the last ingested record when the active instance fails. How is it able to identify the correct entry data point to continue the ingestion for a zero-data-loss purpose?	**For streaming extractors (i.e. IP21 and PI):** In HA run duplicate, shared nothing instances on separate VMs. CDF handles conflict resolution via the data points upsert mechanism. ● Scenario 1 - Multiple streaming extractors running ○ Multiple data streams will be sent to CDF. CDF will handle conflict resolution via the data points upsert mechanism ● Scenario 2 - One extractor (in HA set) fails ○ CDF will continue to ingest data points without loss ● Scenario 3 - Failed extractor (part of HA set) brought back up ○ CDF will continue to ingest data points without loss, duplicates will be handled through the CDF conflict resolution - data points upsert mechanism **For batch extractors (i.e. most integrations):** Depending on the use-cases these are (typically) not time sensitive in the same way as streaming extracts so an HA setup can have the following characteristics ● Hot/cold via VM migration/duplicate VM ● Hot/cold shared nothing (fully isolated) ● Hot/hot shared nothing (fully isolated) Batch extractors will start automated backfill when bought online after a failure or migration. Such backfill job will run in parallel (lower priority) with rea thel time streaming job. Further Cognite would recommend that the operational protocol including change/patch roll-out/etc is aligned with the identified HA/business requirements (example can be ring-based approach to system/OS patching). **Answer to further clarifications** That is correct, the focus on the reply was high-availability (active/active for streaming). For the scenario where one is running active/passive (hot/cold) the applicable behaviour would be similar to Streaming Scenario 3 - Once Active/Hot extractor goes down NOC infrastructure monitoring will detect host/extractor state through internal tooling or via alert from Cognite Console. Such will trigger a "shutdown" of the active VM/host and bring up the passive/cold extractor node. This

	new extractor node will 1) start streaming datapoints and 2) initiate backfill from source (for example PI). The system will be back in a healthy state and alerts cleared. Depending on the operational pattern the new running extractor can be promoted to primary with the "failed" extractor node classified as secondary (cold). In order to avoid zero-data-loss, the new extractor instance will start somewhat before the old one went down. If a datapoint for an identified time series has already been written, it will simply overwrite in CDF and no duplication is done
Further clarifications v2 Detail how CDF is able to manage data duplication for each of the following High Availability scenarios: ● Active-Active approach: each extractor has more instances running at the same time, sending the same data to CDF more times; ● Active-Passive approach: in some cases (e.g., misconfigurations, orchestration errors, etc.) both Active and Passive could send to CDF the same data	**Answer to further clarifications v2** Regardless of the approach and the orchestration, CDF uses external ids to handle upserts (that is, ids that are calculated based on source system unique keys by the extractor). Thus no data will be duplicated in CDF if it gets pushed several times, and the last instance to reach the queue will define the data content.
Detail what are the HA SLAs of the whole E2E solution (i.e., on-prem components and cloud components) instead of the SLAs of the single GCP component (e.g., Compute Engine and GKE, Cloud Storage, etc.). The availability of exposed services depends on how the whole solution is designed and configured / developed	CDF service availability is specified in the SLA document, this also includes relevant performance details. GCP SLAs were added to document that the availability of CDF does not exceed the SLAs of the underlying services it relies on. Also showing insight into zone/region availability for the relevant GCP services.. Updated Cognite SLA with availability and response has been shared. With regards to the on-prem infrastructure it is not possible for Cognite as this point to give any availability details as this is highly dependent on factors that Cognite does not have insight into nor control over (examples include VM configuration, environment redundancy, networking topology and resilience for LAN/WAN, and internal operations availability and on-call capacity). The Cognite extractors are built- and support HA configurations.
Detail how data retention and data backup frequency detailed in 2.1.1.1 for each of the components of the overall system can be configured and tuned in order to meet business related requirements.	Default SLA defines the backup frequency and retention. Any changes to this will need to be initiated as a commercial conversation (change order) with the Cognite account team. At this point the business related requirements are unclear so not possible to comment on outside the commitment to support the metrics and details from the SLA.

Referring to the "Service features" section, please specify what are the features really used on the current solution instead of all the features provided by the GCP services	Please see the updated SLA document on the core resource types. A further mapping should be part of the use-case documentation (individual + aggregated). The listed services are CDF underlying services and their HA components.

2.2.2 Disaster Recovery (DR)

Disaster Recovery is a subset of Business Continuity Planning and is based on an impact analysis that defines the recovery time objective (RTO) and recovery point objective (RPO).

The applicable metrics (RTO and RPO) are stated in the sections below.

Patterns for DR are divided into three different options; Cold, Warm and Hot. The below DR scenarios are based on the Warm pattern, where the necessary resources are allocated for bringing services back up within the set RTO. Exemptions where Cold or Hot patterns are chosen are explicitly called out.

- Cold: DR resources are reserved. They need to be allocated and deployed to mitigate issues.
- Warm: DR resources are allocated and need to be deployed to mitigate issues.
- Hot: DR resources are deployed. Production and DR environments have a level of dependencies that allows services to run with minimal impact until issues are mitigated.

The continuous deployment (CD) environment and artifacts are hosted in a location that ensures that they are available and operational in the event of a disaster.

Security in the DR environment is configured in the same manner as the production environment. DR security is verified and ensured through testing, monitoring of policies and infrastructure as code.

2.2.2.1 Cloud platform DR

Cloud provider

The provided information is a reference for Google Cloud Platform (GCP), retrieved from GCP public documentation (i.e. https://cloud.google.com, https://cloud.google.com/products, and https://cloud.google.com/terms/sla). Further details and updates need to be retrieved from the GCP SLA site.

Further the list below reflects a summary of services, it is not a 100% list and the accuracy of the provided information/details is based on this and further dependent on delta between

2020-MAY-12 and current date. Where necessary additional details/information should be part of the use-case documentation.

Requirements
The following requirements are covered by the provider:

- Capacity: securing enough resources to scale as needed.
- Security: providing physical security to protect assets.
- Network infrastructure: including software components such as firewalls and load balancers.
- Support: making available skilled technicians to perform maintenance and to address issues.
- Bandwidth: planning suitable bandwidth for peak load.
- Facilities: ensuring physical infrastructure, including equipment and power.

The above requirements are covered by the following (cloud provider) features to implement and perform DR:

- A global network. Google has one of the largest and most advanced computer networks in the world. The Google backbone network uses advanced software-defined networking and edge-caching services to deliver fast, consistent, and scalable performance.
- Redundancy. Multiple points of presence (PoPs) across the globe mean strong redundancy. Your data is mirrored automatically across storage devices in multiple locations.
- Scalability. Google Cloud is designed to scale like other Google products (for example, search and Gmail), even when you experience a huge traffic spike. Managed services such as App Engine, Compute Engine autoscalers, and Datastore give you automatic scaling that enables your application to grow and shrink as needed.
- Security. The Google security model is built on over 15 years of experience with helping to keep customers safe on Google applications like Gmail and G Suite. In addition, the site reliability engineering teams at Google help ensure high availability and prevent abuse of platform resources.
- Compliance. Google undergoes regular independent third-party audits to verify that Google Cloud is in alignment with security, privacy, and compliance regulations and best practices. Google Cloud complies with certifications such as ISO 27001, SOC 2/3, and PCI DSS 3.0.

Cognite
Requirements

The following requirements are covered by Cognite:

- Security: The capacity to safeguard assets and their integrity.
- Robustness: The capacity to remain healthy and uncompromised.
- Resilience: The capacity of services to react and recover quickly.
- Responsiveness: The capacity to absorb disturbances while maintaining functionality.
- Flexibility: The capacity to adapt easily to new requirements.

The above requirements are covered by the following (Cognite) capacities to implement and perform DR:

- Incident handling process
- 24x7 support and engineering on-call rotation.
- People allocation and resource prioritization.
- Allocation of DR resources in alternative cloud environment location
    - Where applicable and aligned with data owner.
- Infrastructure as code.
- Separation of CD and artifact hosting from production environment.
- Multi-regional data backup (where not provided by default via cloud offering)
- Business Continuity Plans based on scenarios in the main categories of:
    - Loss of infrastructure
    - DDos Attack
    - Cognite bug
    - Cognite bug (data corruption)
    - Malicious user
    - User error
    - Loss of connectivity to operator site
    - Data source schema change
- Scheduled testing of DR and Business Continuity Plans.

2.2.2.2 On-premises components DR

Zero-data-loss constitutes an RPO of 0. In this context DR is divided into three patterns; Cold, Warm and Hot.

Cold: Data is not cached nor stored and on-premises components have no redundancy. Recovery is performed by backfilling from the source system.
Warm: Data is cached or stored and on-premises components have active/passive redundancy. Recovery is performed by ingesting cached/stored data post failover.
Hot: Data is cached or stored and there is active/active on-premises components redundancy. Cached/stored data is utilized to handle degraded performance until mitigation.

The extractor HA recommended implementation is as follows

- Streaming extractors
    - In scope: IP21 and PI
    - Run active/active in fully redundant environments with "no shared" infrastructure - ideally separate VMs, separate physical network devices, separate networks/VLAN, separate WAN gateways.
        - Depending on the resilience and availability target.
    - Duplicate datapoints conflict/resolution will be handled through the CDF datapoints upsert mechanism.
- Batch extractors
    - In scope: All other extractors.
    - Can run duplicate active/active but such is not recommended due to the lower sensitivity on "real-time" dataflow.
    - Recommended setup is hot/cold where a second extractor is available (deployed) on isolated/redundant infrastructure.
        - Testing of such cold extractor should happen on a regular basis to confirm a healthy state.

2.3 CORE TOPIC 7 - PERFORMANCE ASSURANCE WITH INCREASING WORKLOAD

2.3.1 Scalability Requirements and Performance measurement

Clarifications	Response
Detail the scaleup capabilities of CDF and of all the components used for custom developments to tackle an increasing load volume	Both CDF and the project/custom infrastructure are built on "elastic infrastructure" allowing it to scale both upwards and downwards with the applied load on the system. I.e. we can say that the systems "autoscale". The cloud layers use primarily the same mechanisms for scaling as for HA (topic 6). In short, horizontal scaling by adjusting the number of instances for a given component. There is also an element of "vertical scaling" by configuring the capacity per unit/instance of a module--but when the system adjusts its total capacity, it primarily does so by adjusting the number of instances. CDF (the product) autoscales both compute capacity and storage capacity (both volume and throughput). The project components primarily handle compute-bound workloads and scales along the compute dimension.
Further clarifications Detail how custom components effectively scale up/down (e.g., kubernetes autoscaling)	**Answer to further clarifications** .Cloud Run autoscaling: https://cloud.google.com/run/docs/about-instance-autoscaling Cloud Functions scaling: https://cloud.google.com/functions/docs/max-instances Kubernetes auto-scaler: https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-autoscaler..

2.3.1.1 On premise components requisites base sizing

The main on-premises components are the extractors. Extractors are covered in 1.1.1.1. and 1.1.2.1. Individual, minimum requirements per extractor type is presented in the extractor product documentation. In general:

- Extractors run on WinServer 2016+
- Min HW: 4x CPu, 16GB ram
- 200 GB local data disk

- Network connectivity to the desired source systems
    - FW openings to sources depend on the sources systems themselves.
- Network connectivity to the target system
    - CDF: api.cognitedata.com, https (443)
    - While Cognite are monitoring the extractors: prometheus-push.cognite.ai, https (443)

Clarifications	**Response**
Detail which are the suggested specs for a single machine containing all the extractors	Nearly impossible to spec this up-front. It is a much safer path to do the following: • Separate streaming extractors (i.e. IP21) and the batch extractors on separate machines. • Start with the baseline HW config (see above) and measure the load on the HW for each additional extractor applied to the machine. • The extractor HW will primarily be constrained on network I/O and CPU. When you observe a steady state CPU consumption of >70% it is time to increase the HW capacity.

2.3.1.2 Performance Metrics

The extractors themselves do not autoscale in an on-premises setup.

2.3.2 Performance assurance

2.3.2.1 On premise components

The extractors emit logs and metrics that can be captured via log scraping and/or Prometheus. The metrics include indicators on resource consumption (CPU, RAM, etc.) productivity (flow rates, total data objects processed, stream latency etc.) and stability (reconnects, source error codes, etc.). The exact metrics depend on the extractor.

You would set metric thresholds in the monitoring system (i.e. Prometheus or your monitoring system of choice).

Scaling of extractors can be done in two ways:

1. Vertical scaling: Increasing the compute power of the node hosting the extractor.

2. Horizontal scaling: Splitting the work performed by the extractor across multiples instances.

We recommend scaling by option 1 as this is the easiest to manage. Also, we have not yet come across examples where option 1 was a bottleneck when dealing with on-premises sources.

Option 2 is an advanced configuration requiring deep knowledge of the source system behavior. It is based on sharding the set of work between multiple instances.

Clarifications	**Response**
Based on your experience and understanding of the client context, detail what are the suggested extractors scaleup mechanisms alternatives	We recommend vertical scaling for the extractors. That is, a single source data set is handled by a single extractor instance. If you need more capacity, then increase the HW capacity of the node hosting the extractor. In your setup, we think you can host multiple batch extractor instances on the same node. Given this setup, you can scale up along the following: • Increase the HW capacity of the node. • Add more HW nodes and redistribute the extractor instances across more nodes.

2.3.2.2 Cloud components

CDF is fully monitored and operated by Cognite. We continuously monitor the health and behavior of the various CDF services to ensure the best possible service level experience.

For the custom cloud components used for the project delivery we will use Google Cloud (logging and) Monitoring. It is pre-wired into all the cloud infrastructure and allows for easy design, collection, visualization and alerting of metrics.

The cloud layers use primarily the same mechanisms for scaling as for HA (topic 6). In short, horizontal scaling by adjusting the number of instances for a given component. There is also an element of "vertical scaling" by configuring the capacity per unit/instance of a module--but when the system adjusts its total capacity, it primarily does so by adjusting the number of instances.

Scaling is performed using the native capabilities of the cloud infrastructure:

- GKE autoscaler for Kubernetes. See 2.2.1.1.
- Cloud Run: container instance autoscaling: https://cloud.google.com/run/docs/about-instance-autoscaling
- Cloud functions: instance autoscaling: https://cloud.google.com/functions/docs/max-instances

3 SECURITY

3.1 CORE TOPIC 8 - SECURITY BY DESIGN

3.1.1 Security Operations & Monitoring

THE COGNITE SECURITY PROGRAM IS BUILT ON THE FOLLOWING CORE PRINCIPLES

SHALL BE SECURE
The capacity to safeguard assets and their integrity.

SHALL BE ROBUST
The capacity to remain healthy and uncompromised.

SHALL BE RESILIENT
The capacity of services to react and recover quickly.

SHALL BE RESPONSIVE
The capacity to absorb disturbances while maintaining functionality.

SHALL BE FLEXIBLE
The capacity to adapt easily to new requirements.

The role as a Trusted Custodian is at the core of our DNA. In order to safeguard the confidentiality, integrity and availability of our customers' data, Cognite is fully committed to maintaining the highest standards in security and quality through continuous improvement of our Management System.

The inputs to the security program include (but not limited to)

- Standards requirements
- Data-owner requirements
- Legal and regulatory requirements

Secure and Resilient by Design

- Strong security and quality GRC (management) program

    - Built on NIST CSF
    - Certified against ISO 27001 and 9001
- Secure development and operations
- Secure Software Development Lifecycle
- Encryption by default
- Data Governance and Access Control
- Data Owner Insight - Observability and Audit

System security standards
Cognite currently hold ISO 9001:2015 and ISO 27001:2013/2017 certifications

In addition to the in place ISO certifications Cognite is using relevant (curated set of) controls from ISO 27018, CIS, NIST 800-53, NIST 800-171, and ENISA.

Google Cloud supports ISO 27001, ISO 27017, ISO 27018, SOC 1, SOC 2, SOC 3, CSA STAR, FedRAMP, Sarbanes-Oxley (SOX), EU Data Protection Directive, EU Model Contract Clauses, GDPR and a number of other standards listed at
https://cloud.google.com/security/compliance/

For a full list of Google Cloud Platform standards and certifications see:
https://cloud.google.com/security/compliance/
In addition: Google has a dedicated internal audit team that reviews compliance with security laws and regulations around the world. As new auditing standards are created, the internal audit team determines what controls, processes, and systems are needed to meet them. This team facilitates and supports independent audits and assessments by third parties.

Security - Secure Development and Operations
- Secure development and operations
    - All code version controlled and Peer-Review enforced
    - Infrastructure as code (Terraform) to ensure integrity and avoid drift of Service Fabric
    - Vulnerability scanning and validation through the software pipeline
        - Image Vulnerability & Dependency and license scanning
        - Static Code Analysis & Fuzzing
    - Behavioural analytics
        - GCP CSCC
        - Cognite Security Insights (CSI)

    - Threat modeling based on the MSFT STRIDE model
    - Service Maturity and Error Budget
- Two access levels for Cognite employees
    - Default data access is managed through data owner directory using established onboarding/offboarding processes
    - Privilege access for infrastructure and security engineers.
        - Data owner will have access to list of such identities
        - Individual NDAs will be signed as per data owner requirement and policy

Security Controls (technical and infrastructure)

Services that are provided via the Cognite API (Layer 7 RESTful Web Service) are scanned and monitored. Cognite Data Fusion is protected by several defensive layers including scanning of-, and restriction on traffic entering and leaving- global perimeters as well as inside the clusters and between services.

Technical and infrastructure security controls currently in place:

- Secure Low Level Infrastructure:
    - Security of physical infrastructure
    - Hardware design and provenance
    - Secure boot stack and machine identity
- Infrastructure as code
    - Infrastructure (service fabric) is handled as code and managed through Terraform to ensure that any and all changes are peer reviewed, approved, and tracked aligned with Cognite Secure SDLC practices.
- Cloud Armor
    - DDos Protection
    - Input Validation
    - Scrubbed HTTP return values
    - Logging
    - Error handling
    - Throttling
    - WAF
    - Application Defense
- Cloud Security Command Center
    - View and monitor an inventory of cloud assets
    - Scan storage systems for sensitive data
    - Cloud Anomaly Detection

    - Detect common web vulnerabilities
    - Review access rights to your critical resources
- IAP
    - Secure access to services and systems
- Secure Service Deployment
    - Service Identity, Integrity, and Isolation
    - Inter-Service Access Management
    - Encryption of Inter-Service Communication
- User Identity
    - Authentication
    - Authorization
    - Login Abuse Protection
- Storage Services
    - Deletion of Data
    - Encryption at rest
- Operational Security
    - Intrusion Detection
    - Reducing Insider Risk
    - Safe Employees and Credentials
    - Safe Software Development
    - Integrate with customer user directory for authentication and authorization
    - Encryption in transit (all flows)
    - Encryption at rest (all data/information)
    - Code analysis
    - Peer-review and peer-approval (code changes)
    - Unit- and integration tests
    - Dev - Test - Prod environment (including manual gating to production)
    - Code coverage monitoring
    - Service maturity policy/matrix and monitoring
- Process based Security Controls
    - Threat Modeling
    - Vulnerability Management
- Testing methodology and tools
    - OWASP
    - kube-bench
    - kube-hunter
    - Burp Suite Professional
- Forseti Security

    - Inventory, policy monitoring, rules enforcement, and access insight.
- RedLock
    - Compliance assurance, security governance, threat detection, vulnerability management, application profiling, auto-remediation, incident investigation
- Checkmarx
    - Static application security testing, open source analysis

Cognite Secure Software Development Lifecycle

Cognite Security Development Lifecycle (SDL) is a fundamental part of the Cognite SDLC practices.

Secure SDLC practices is an established methodology and process to ensure a systematic and secure software development lifecycle (SSDL).

Phase 1 - Training

- Security awareness sessions — monthly security awareness sessions on topics like phishing, safe behavior online, password management, and endpoint security.
- Monthly phishing simulations - leveraged to measure effectiveness of and plan awareness training.
- Security training and deep dives — scheduled, upon request by teams, or when introducing a new technology, process, or methodology with impact on security. These include training on threat modeling, identity and access management, secrets management, and others.

Phase 2 - Requirements

- Security requirements — design documents including security requirements are written for new systems and components and are reviewed and discussed in the Architecture Forum meeting place, where the security team is present and assists in identifying and clarifying security requirements.

Phase 3 - Design

- Threat modeling — for specific features/components/systems with security impact, a threat model is done with assistance from the security team when needed. Security team reviews threat models.

Phase 4 - Implementation

- Version Control System (VCS) — all code, including application source code, infrastructure as code, configuration, pipeline definitions, and manifests resides in

VCS to ensure change and release management while also providing an auditable change log.

- Code review — all code in VCS is required to be peer reviewed and approved before being merged from a feature branch into a master (production) branch.
- Automated build pipeline — building executable programs, automated quality control gates including functional, integration, and security testing, building of container images, and deployment are handled by automated build pipelines in a build server triggered by changes pushed to VCS. This allows for consistent and repeatable build steps with quality and security control gates.
- Static Application Security Testing — Security testing with reporting, part of CI/CD pipeline, prior to being approved for production release.
- Software Composition Analysis — Security and licenses scanning with alerts for vulnerable dependencies and use of non-compliant licenses.

Phase 5 - Verification

- Vulnerability Management — Cognite is using several tools to continuously scan for and identify vulnerabilities. Some of the tools currently being used include Google Container Registry Container Analysis, GitHub security alerts, kube-bench, and kube-hunter. Vulnerability management covers activities like network scanning, container scanning, dynamic web application scanning. Detected vulnerabilities are evaluated, ranked, and the mitigation is agreed upon with developers. Vulnerability (issue/bug) is tracked and kept open until mitigated.
- Configuration Management — Infrastructure and configurations are managed as code and this allows for continuous validation of running configuration vs. approved/defined configuration. This protects against platform/environment drift. Changes to platform/environment is subject to test and validation before being rolled out to production.
- Penetration testing — Cognite executes regular security and penetration testing of Cognite Data Fusion services and components.

Phase 6 - Release

- Automated deployments — Automated deployments to Google managed Kubernetes clusters (GKE). Remove the need for developer access to production systems and provide consistent repeatable deployments with rollback capabilities.
- Incident response plan — Cognite does have an incident response plan in place that documents the critical service details, defines playbook requirements, and contact/escalation steps. Services are mapped to- and requirements are defined by

the service maturity policy/matrix. The Cognite Service Reliability team is monitoring and ensuring that services are in compliance with the defined requirements.

Phase 7 - Maintenance

- Patch management — Services are redeployed when an update version of the image, service, or dependencies in the container is needed. Google Kubernetes Engine underlying infrastructure is managed and kept up-to-date by Google.
- Logging — Cognite and Google Cloud Platform offers an audit trail of actions when Cognite- and/or Google Cloud personnel interact with your data or related infrastructure. The Audit Trail builds on already robust controls that restrict administrator activity to actions only with valid business justification, such as responding to a specific ticket our customers have initiated or recovering from an outage.
- Security Incident Response — Cognite does have an incident response plan that includes detection and response to incidents. Incidents above agreed threshold and/or impact will be reported through initial report followed by detailed customer postmortem.
- Runtime container security and intrusion detection — Using several tools including GCP native, open source, and commercial tooling/services from Palo Alto Networks.
- Misconfiguration detection and compliance — Using GCP native services, Palo Alto Networks/RedLock, and Forseti Security to ensure continuous scanning and mapping of configurations against standards and best practices.
- Reporting — Monthly security reports are generated and shared with the Cognite organization and senior leadership (including Board of Directors). Where applicable, or requested, relevant information will be shared with external parties (including customers and partners). This will be in addition to the already/contractually committed external reporting cycle.

Security - Data Owner Insight

- Activities are logged and posted to the following Cognite internal targets
    - Insight and Telemetry
    - Audit
- Insight and Telemetry
    - Used to continuously monitor and operate service
        - Logs
        - Metrics
        - Traces
- Audit

- Immutable audit trail for customer data and related infrastructure activities
- Isolated storage
- Restricted access
- Audit logs be made available to customer for ingestion into own SIEM

Logging and Incident (overview)
Cognite proactively and continuously monitors infrastructure, services, accounts, and logs for vulnerabilities and irregular activities.

For the continuous monitoring and testing Cognite is using 3rd party tools as well as custom-built tooling and solutions. Services will be measured against the service maturity framework (BSIMM equivalent) where requirements for observability (including monitoring, logging, tracing, alerts) are tracked.

Services not meeting the required threshold will not be approved for production. The maturity measurement also measures service optimization and monitoring of change in service behavior or resource consumption. Security posture of services are measured through tools and metrics.

Incidents are called based on criterias (including but not limited to) observability insight, vulnerabilities identified, and notifications/ticket. Incident calling is an established (and automated) process.

Observability (metrics/logs/tracing)
Cognite is operating a full observability stack using GCP native/managed tools, open source components, and custom code/tools.

Metrics
Default using Stackdriver (GCP native) and Prometheus (internally hosted). Prometheus will most likely be replaced with VictoraMetrics due to performance and scaling limitations with Prometheus.

Logs (application/non-sensitive/non-audit)
Default shipped to managed GCP service Stackdriver using native transport architecture including fluentd.

Audit Logs

Cognite-managed logging mechanism transporting logs from containers/pods using a custom sidecar and transport to BigQuery. Such logs are not sent to Stackdriver due to the sensitivity of content and limitations/complexity in Stackdriver RBAC. Such logs are not using fluentd due to limitations (performance and reliability).

Tracing (application insight/non-sensitive/non-audit)
Cognite distributed tracing is instrumented OpenTracing (evolving into OpenTelemetry) project/"standard" and implemented with SaaS service LightStep. Code instrumented will send trace data to satellites that will then pass on to the central service. LightStep is a tracing service that supports sampling-free tracing resulting in high quality/high value insights.

Data-owner observability insights (for end-to-end analysis)
Cognite supports integration with/export to data-owner SIEM/security stack for log analysis enabling data-owner to have end-to-end visibility and ability to run independent verifications and have own integrity controls (end-to-end). Further data-owners can use Cognite Console to enable data quality monitoring with rules and alerts on data behaviour.

Incident Response (overview)
The Cognite Incident Response Plan includes details on detect and respond. All incidents above the agreed threshold and/or involving customer data will be reported.

Initial report immediately after detection then complete report after postmortem. Postmortem to be completed within 3 working days. The Customer will be able to respond to the report, service provider will then follow up and acknowledge within five working days.

Incident handling at Cognite is built on the NIST SP 800-61 R2 guide and certified against ISO 9001:2015 and 27001:2013/2018.

Incident Response (phases)

- Phase -1
    - Preparation (including training/awareness)
- Phase 0
    - Calling incident - Internal tooling for calling incident (including type), invite stakeholder, create communication channels, and postmortem documents
- Phase 1
    - Identification and Triage (situational awareness)
- Phase 2

    - Containment
- Phase 3
    - Eradication
- Phase 4
    - Recovery and Post-Incident Activity

Security program related to "software, service, and product assurance"
In the software, service, and product assurance program there are two primary layers

Internal layer - Operated by Cognite using Cognite personnel and internal/open-source/paid software/services and tools
External layer using 3rd party expertise/specialists

High-level overview internal layer
The internal layer is the core of the security program where internal security capability and capacity is paired with internal systems understanding and insights to continuously test and validate product and system security. This program is mainly manifested through the Cognite Secure SDLC, starting with the source code and "ending" with continuous monitoring of the running "binary".

From the development process point of view it is important to highlight SDLC components like

- Threat Modeling/Architectural Risk Analysis
- Peer-Review
- Static Analysis
- Dependency and License
- Automated testing (for example fuzzing)
- Manual expert testing (pen-testing)

For internal pen-testing there is a continuous program where components and/or end-to-end services are subject to offensive/red-team activities. The trigger for such activities can be time since-last-activity or volume/complexity of change. The testing methodology ranges based on target.

High-level overview external layer
Cognite will use 3rd party security specialists and auditors to do a complete test, evaluation, and audit of the environment on a 12-month cycle. Below are some highlights of the external layer activities.

External VAPT at least every 12 months
ISO 9001 and 27001

- Certification audit completed and passed December 2019
- First surveillance audit Q4 2020

SOC 2

- Type 1 planned end-of-year 2020
- Type 2 planned Q3 2021 (Type 1 + 6 months)

If significant issues or irregularities are detected inside the 12-month cycle it will be adjusted and external party will complete an immediate test, evaluation, and audit of identified issue/irregularity.Follow-up reply/clarification

<table>
<tr><th>Clarifications</th><th>Response</th></tr>
<tr><td>Define better and clearly delineate the shared responsibility between Google and Cognite;</td><td>Shared Responsibility Matrix cloud (overview):
<table>
<tr><th></th><th>IaaS</th><th>PaaS</th><th>SaaS</th></tr>
<tr><td>Content</td><td>Cognite</td><td>Cognite</td><td>Cognite</td></tr>
<tr><td>Access Policies</td><td>Cognite</td><td>Cognite</td><td>Cognite</td></tr>
<tr><td>Usage</td><td>Cognite</td><td>Cognite</td><td>Cognite</td></tr>
<tr><td>Deployment</td><td>Cognite</td><td>Cognite</td><td>Google</td></tr>
<tr><td>Web application security</td><td>Cognite</td><td>Cognite</td><td>Google</td></tr>
<tr><td>Identity</td><td>Cognite</td><td>Google</td><td>Google</td></tr>
<tr><td>Operations</td><td>Cognite</td><td>Google</td><td>Google</td></tr>
<tr><td>Access and authentication</td><td>Cognite</td><td>Google</td><td>Google</td></tr>
<tr><td>Network security</td><td>Cognite</td><td>Google</td><td>Google</td></tr>
<tr><td>Guest OS, data & content</td><td>Cognite</td><td>Google</td><td>Google</td></tr>
<tr><td>Audit logging</td><td>Google</td><td>Google</td><td>Google</td></tr>
<tr><td>Network</td><td>Google</td><td>Google</td><td>Google</td></tr>
<tr><td>Storage + encryption</td><td>Google</td><td>Google</td><td>Google</td></tr>
</table></td></tr>
</table>

Hardened Kernel + IPC	Google	Google	Google
Boot	Google	Google	Google
Hardware	Google	Google	Google

Cognite is mainly using managed services and where possible using the underlying GCP functionality related to operations and security. Below is an overview of the security functionality/services available in GCP.

Overview Google infrastructure security services:

Usage	• Audit Logging • Safe Browsing API • BeyondCorp • Security Key Enforcement
Operations	• Compliance & Certifications • Live Migration Infra maintenance & patching • Threat analysis and intelligence • Open Source Forensic tools • Anomaly Detection (infrastructure) • Incident Response (Infrastructure)
Deployment	• Google Services TLS encryption with perfect forward secrecy • Certificate Authority • Free and automatic certificates • DDoS Mitigation (PaaS & SaaS)
Application	• Peer code review & Static Analysis (Infrastructure SDLC) • Source code provenance (Infrastructure) • Binary Verification (Infrastructure code) • WAF (PaaS & SaaS Use cases) • IDS/IPS (PaaS & SaaS Use cases) • Web Application Scanner (Google Services)
Network	• Infrastructure RPC encryption in transit between data centres • DNS • Global Private Network • Andromeda SDN Controller • Jupiter Datacenter Network • B4 SDN Network
Storage	• Encryption at rest • Logging • Identity and Access Management • Global at scale Key Management Service

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>OS + IPC</td><td>• Hardened KVM Hypervisor
• Authentication for each host and each job
• Curated Host Images
• Encryption of Interservice Communication</td></tr>
<tr><td>Boot</td><td>• Trusted Boot
• Cryptographic Credentials</td></tr>
<tr><td>Hardware</td><td>• Purpose-built Chips
• Purpose-built Servers
• Purpose-built Storage
• Purpose-built Network
• Purpose-built Data Centers</td></tr>
</table>
</td>
</tr>
<tr>
<td>Referring to the section "Cognite SSDLC" provide a more detailed description of the solution.</td>
<td>Unclear on the exact clarification requested so providing some additional details on the Cognite Secure Development Lifecycle. The Cognite SSDCL is in essence a series of steps/phases for software development in Cognite. The SDLC is a living framework covering areas from people and training through developing, testing, deploying, operating, maintaining, and correcting software and services.

Security testing and validation is part of the SDLC/entire product life cycle (from source to running service) and include the following activities:

Phase 1 - Training

• Onboarding security awareness training
• Security awareness sessions — monthly security awareness sessions on topics like phishing, safe behavior online, password management, and endpoint security.
• Phishing simulations - leveraged to measure effectiveness of and plan awareness training.
• Security training and deep dives — scheduled, upon request by teams, or when introducing a new technology, process, or methodology with impact on security. These include training on threat modeling, identity and access management, secrets management, and others.

Phase 2 - Requirements
Security requirements — design documents including security requirements are written for new systems and components and are reviewed and discussed in the Architecture Forum meeting place, where the security team is present and assists in identifying and clarifying security requirements.

Phase 3 - Design
• Threat modeling — for specific features/components/systems</td>
</tr>
</table>

	with security impact, a threat model is done with assistance from the security team when needed. Security team reviews threat models. ● Architecture Risk Review ● Design Review **Phase 4 - Implementation** ● Version Control System (VCS) — all code, including application source code, infrastructure as code, configuration, pipeline definitions, and manifests resides in VCS to ensure change and release management while also providing an auditable change log. ● Code review — all code in VCS is required to be peer reviewed and approved before being merged from a feature branch into a master (production) branch. ○ Enforced by branch protection rules on code repositories. ● Automated build pipeline — building executable programs, automated quality control gates including functional, integration, and security testing, building of container images, and deployment are handled by automated build pipelines in a build server triggered by changes pushed to VCS. This allows for consistent and repeatable build steps with quality and security control gates. ● Static Application Security Testing — Security testing with reporting, part of CI/CD pipeline, prior to being approved for production release. SAST configured to stop the build pipeline or to automatically create Jira (issue tracker) issues. SAST tools invoked via GitHub webhooks or in Jenskinsfile build pipeline definitions: ○ Checkmarx CxSAST ○ Spotbugs ○ PMD ● Software Composition Analysis — Security and licenses scanning with alerts for vulnerable dependencies and use of non-compliant licenses. ○ Checkmarx CxOSA ○ GitHub Security Alerts + Dependabot ● Unit Testing ● Integration Testing **Phase 5 - Verification** ● Vulnerability Management — Cognite is using several tools to continuously scan for and identify vulnerabilities. Some of the tools currently being used include Google Container Registry Container Analysis, GitHub security alerts, kube-bench, and

	kube-hunter. Vulnerability management covers activities like network scanning, container scanning, dynamic web application scanning. Detected vulnerabilities are evaluated, ranked, and the mitigation is agreed upon with developers. Vulnerability (issue/bug) is tracked and kept open until mitigated.
	
	• Configuration Management — Infrastructure and configurations are managed as code and this allows for continuous validation of running configuration vs. approved/defined configuration. This protects against platform/environment drift. Changes to platform/environment is subject to test and validation before being rolled out to production.
	
	• Penetration testing — Cognite executes regular security and penetration testing of Cognite Data Fusion services and components.
	
	• Dynamic Application Security testing, both as part of regular penetration testing and regular automated tests with Google Cloud Security Command Center Web Security Scanner.
	
	• Image Vulnerability Scanning
	  ○ Container Analysis
	• Fuzzing
	  ○ Custom
	  ○ AFL
	• Network Vulnerability Scanning with Tenable.io
	  ○ Daily scans of publicly exposed interfaces
	
	Phase 6 - Release
	Automated deployments — Automated deployments to Google managed Kubernetes clusters (GKE). Remove the need for developer access to production systems and provide consistent repeatable deployments with rollback capabilities.
	Incident response plan — Cognite does have an incident response plan in place that documents the critical service details, defines playbook requirements, and contact/escalation steps. Services are mapped to- and requirements are defined by the service maturity policy/matrix. The Cognite Service Reliability team is monitoring and ensuring that services are in compliance with the defined requirements.
	
	Phase 7 - Maintenance
	Patch management — Services are redeployed when an update version of the image, service, or dependencies in the container is needed. Google Kubernetes Engine underlying infrastructure is managed and kept up-to-date by Google.
	
	Logging — Cognite and Google Cloud Platform offers an audit trail of actions when Cognite- and/or Google Cloud personnel interact with your data or related infrastructure. The Audit Trail builds on already robust

	controls that restrict administrator activity to actions only with valid business justification, such as responding to a specific ticket our customers have initiated or recovering from an outage. Security Incident Response — Cognite does have an incident response plan that includes detection and response to incidents. Incidents above agreed threshold and/or impact will be reported through initial report followed by detailed customer postmortem. Runtime container security and intrusion detection — Using several tools including GCP native, open source, and commercial tooling/services from Palo Alto Networks. Reporting — Monthly security reports are generated and shared with the Cognite organization and senior leadership (including Board of Directors). Where applicable, or requested, relevant information will be shared with external parties (including customers and partners). This will be in addition to the already/contractually committed external reporting cycle. Misconfiguration detection and compliance — Using GCP Cloud Security Command Center to ensure continuous scanning and mapping of configurations against standards and best practices: ● Continuous asset discovery and inventory ● Security Health Analytics ○ Exposed assets ○ Asset configurations ○ IAM configurations ● Event threat detection (log based) ○ Malware ○ Cryptomining ○ Brute force SSH ○ Outgoing DoS ○ IAM audit ● Container threat detection ○ Suspicious binary ○ Suspicious library ○ Reverse shell ● Security health analytics ○ CIS 1.0 ○ PCI DSS v3.2 ○ NIST 800-53 ○ ISO 27001 ● Web security scanner ○ XSS ○ Flash injection ○ Mixed-content ○ Clear text passwords ○ Usage of insecure JavaScript libraries

Logging. Detail how security logs (e.g., identity of the API caller, time of the API call, source of the API caller, the request parameters, etc.) can be sent to the NOC Security Operations Center for further analysis as part of NOC existing processes. Provide further details how COgnite logging system (Google Stackdriver) can integrate with NOC on-prem SIEM (LogRhythm) (e.g., upload of logs to a storage blob, etc.) **Further clarifications** Detail a suggested mechanism to automatically pull logs from Cognite systems before the API Logs are available in Q3 (e.g., provide logs to NOC through a storage service, etc.). Clarify log types available (i.e., we're not referring to the underlying GCP infrastructure but Cognite components logs) **Further clarifications v2** Describe an available / viable process to share with NOC a weekly Audit log report before the Audit log APIs availability	**Audit logging** Audit logs are currently available on a request basis with a plan to make available an Audit Log API by end of Q3 2020. With this API NOC can access the audit logs for insights, analysis, and import to own SOC/SIEM. **Cognite cloud logging** Logging outside of the above mentioned audit logging is not planned to be made available to the data owner. **Answer to further clarifications** As per conversation and agreement in the meeting Cognite will investigate the possibility of enabling temporary access to audit logs through an alternative mechanism. Where such a temporary solution has a negative impact on the delivery of the planned Audit Log API it will be made clear. Cognite understanding is that the shared agreement will then be to prioritize the Audit Log API. Next steps (Cognite to get back to NOC by end day June-4th-2020): ● Cognite identify possibility for temporary audit log access ● Cognite to investigate possibility to adjust timeline for the Audit Log API **Update on CDF Audit Log access** ● Given the sensitivity of the information in scope (audit logs) Cognite will not recommend a temporary solution/workaround due to the risk involved. ● Cognite is recommending and committing to the following ○ Audit Log API to be available end of August 2020 ○ NOC will be offered early (alpha) access to the MVP Audi Log API. This is expected to be available in the first half of August 2020. **Answers to further clarifications v2** In the period until the Audit Log API is available (as per above) Cognite will not have an automated solution with weekly reporting. If such is required this will be scoped (estimated to 2 weeks) with export to a secure GCS bucket where NOC can ingest from. This work will impact the delivery date of the Audit Log API (MVP and v1). There is a commitment, stated earlier in the document and in the CAIQ, to support requests for audit logs but this is not designed to be used as a way to get regular/frequent updates but can be used to request for example one update in the relevant time period (between current date and Audit Log API MVP date).
Security incident response. Provide details about threshold mentioned for incidents reports and further details about the response SLAs, roles and responsibilities etc. in case of a security incident;	**Threshold** Incident involving customer data. **Timelines** Notification shall be issued without delay, no later than 72 hours after detection. Any delays outside the 72 hours commitment is to be accompanied by reasons for delay. The notification should contain (as a minimum) details on the incident, detection, consequences, remediation

	action, and postmortem details. **Roles and Responsibilities:** For security incidents communication will be managed by Cognite security team and part of the incident process. **Request to NOC:** Provide Cognite with NOC security contact(s), Cognite will use the provided information to draft up a standard "Security Rules of Engagement" document.
Further clarifications clarify differences with the other point "Incident Response (overview)" specified later	**Answer to further clarifications** As per the meeting this is the baseline commitment aligned with the requirements in GDPR. The default text in the reply is: Data breach notification Personal Data Breach as described in the GDPR is not relevant as personal data is by default not processed or stored in Cognite Data Fusion. Cognite will however adhere to the principles and guidelines from the GDPR and inform the customer and data owner of any incident involving customer data. Such notification shall be issued without delay, no later than 72 hours after detection. Any delays outside the 72 hours commitment is to be accompanied by reasons for delay. The notification should contain (as a minimum) details on the incident, detection, consequences, remediation action, and postmortem details. The details provided later in the document overrules and are the actual SLA timelines and commitments.
Runtime container security and intrusion detection. Provide more details about this section specifying how the networking controls in place for this purpose will add value to NOC services in scope.	Through shift left activities, adding controls and capabilities closer to the source code, as well as continuous scanning of the running environment (configuration and components) against established best practices Cognite is managing risk through protection and detection capabilities. Configuration control and validation of operating state reduce the attack surface (protection), avoid configuration drift, and ensure strong detection capabilities. These are all key aspects of protecting the NOC data and services in scope. Examples of tools include: ● Google Container Registry ● Container Analysis ● Cloud Security Command Center Misconfiguration detection and compliance — Using GCP Cloud Security Command Center to ensure continuous scanning and mapping of configurations against standards and best practices: ● Continuous asset discovery and inventory ● Security Health Analytics ○ Exposed assets ○ Asset configurations ○ IAM configurations

	• Event threat detection (log based)
	  ○ Malware
	  ○ Cryptomining
	  ○ Brute force SSH
	  ○ Outgoing DoS
	  ○ IAM audit
	• Container threat detection
	  ○ Suspicious binary
	  ○ Suspicious library
	  ○ Reverse shell
	• Security health analytics
	  ○ CIS 1.0
	  ○ PCI DSS v3.2
	  ○ NIST 800-53
	  ○ ISO 27001
	• Web security scanner
	  ○ XSS
	  ○ Flash injection
	  ○ Mixed-content
	  ○ Clear text passwords
	  ○ Usage of insecure JavaScript libraries

Misconfiguration detection and compliance. Provide more details about this section specifying how this will add value to NOC services in scope.	Through shift left activities, adding controls and capabilities closer to the source code, as well as continuous scanning of the running environment (configuration and components) against established best practices Cognite is managing risk through protection and detection capabilities. Configuration control and validation of operating state reduce the attack surface (protection), avoid configuration drift, and ensure strong detection capabilities. These are all key aspects of protecting the NOC data and services in scope.
	
	Example of tool used include:
	• GCP Security Health Analytics
	• Cloud Security Command Center
	
	Misconfiguration detection and compliance — Using GCP Cloud Security Command Center to ensure continuous scanning and mapping of configurations against standards and best practices:
	• Continuous asset discovery and inventory
	• Security Health Analytics
	  ○ Exposed assets
	  ○ Asset configurations
	  ○ IAM configurations
	• Event threat detection (log based)
	  ○ Malware
	  ○ Cryptomining
	  ○ Brute force SSH
	  ○ Outgoing DoS
	  ○ IAM audit
	• Container threat detection
	  ○ Suspicious binary

	○ Suspicious library ○ Reverse shell ● Security health analytics ○ CIS 1.0 ○ PCI DSS v3.2 ○ NIST 800-53 ○ ISO 27001 ● Web security scanner ○ XSS ○ Flash injection ○ Mixed-content ○ Clear text passwords ○ Usage of insecure JavaScript libraries
Reporting. Provide more details about this section specifying how this will add value to NOC services in scope (e.g., customized reporting capabilities on security metrics for NOC, etc.). **Further clarifications** Clarify better the point. It's not clear the following sentence "There should be an expectation that additional security reporting will be made available as part of the subscription levels/tiers"	Currently there is no custom security reporting offered. There should be an expectation that additional security reporting will be made available as part of the subscription levels/tiers. **Answer to further clarifications** Cognite is currently working on offering a tiered subscription model where individual reporting will be part of the higher level tiers. For NOC, based on estimated subscription level, the following should be expected: ● Quarterly security update ● Annual security report ● Annual security review (in-person/remote) Further Cognite Security plan to build and expose a security and health dashboard, the timing of such is yet not finalized but tentative MVP timeline is H2-2021.
Security Organization section in the Appendix of this document since it is not much relevant to an "Architecture" document.	Removed from the submitted material
Observability. Provide details (as requested in the logging chapter) about how it is possible to integrate NOC Security Operations Center and about the possibility to define and agree between Cognite and NOC the metrics and KPIs to keep a watch for.	Outside of the audit logs and security reporting, as per the above, additional observability information is currently not planned to be made available to NOC. The exception to this is where such can be part of the information shared as part of an agreed activity or as part of a postmortem document. For general SLA and availability metrics such will be provided through mechanisms managed by Cognite Support.
Incident Response (overview) –Provide more details about what certifies as an incident, the SLAs, roles and responsibilities from NOC and Cognite and what is Cognite breach notification policy requiring them to notify NOC of suffering a breach of security that	Triggers to declare a Major Incident: In general, one or more of the following should apply: ● Noticeable loss of data or service availability, particularly across multiple services, and not likely attributable to a single bad recent deployment

compromised NOC data;	• Exposure of sensitive data • Unexpected critical/large scale loss of data integrity **Urgency** *Critical* The change is immediately necessary to prevent severe business impact. The damage caused by the Incident increases rapidly and inhibits end-user from completing time sensitive work. Several users with VIP status are affected. *High* The change is needed as soon as possible because of potential damaging service impact. The damage caused by the Incident increases considerably over time. A single user with VIP status is affected. *Medium* The change will solve irritanting problems or repair missing functionality. This change can be scheduled. The damage caused by the Incident only marginally increases over time. Work that cannot be completed by the end-user is not time sensitive. *Low* The change will lead to improvements, changes in workflow, or configurations. This change can be scheduled. The damage caused by the Incident does not increase over time. The Incident does not affect the current work of the end-user. **Impact** *Extensive/ Widespread* Customer's business experiences significant loss or degradation of services resulting in significant financial impact. Alternative functions/workarounds do not exist or cannot be utilized. *Significant / Large* Customer's business has moderate loss or degradation of services but work can reasonably continue in an impaired manner. Alternative functions/workarounds exist and can be utilised. *Moderate / Limited* Customer's business is substantially functioning with minor or no impediments. *Minor / Localized* Customer's business is functioning well and the ticket is a question, feature request or inquiry of administrative nature **Severity and response/resolve time**

Impact / Urgency	Critical	High	Medium	Low
Extensive/Widespread	Sev A	Sev A	Sev A	Sev B
Significant/Large	Sev A	Sev B	Sev B	Sev B
Moderate/Limited	Sev B	Sev C	Sev C	Sev C
Minor/Localized	Sev C	Sev C	Sev C	Sev D

Sev A (Major Incident)

- First response: 10 minutes
- Follow up: 30 minutes / Escalation time: 30 minutes
- Target resolution time: 8 business hours

Sev B

- First response: 1 hour
- Follow up: 4 hours
- Target resolution time: 2 business days

Sev C

- First response: 2 hours
- Follow up: 24 hours
- Target resolution time:5 business days

Sev D

- First response: 4 hours
- Follow up: 72 hours
- Target resolution time: 10 business days

Cognite's responsibility in an incident are scoped to the response related to and recovery of Cognite managed infrastructure, Cognite Data Fusion and mitigation of errors/bugs in Cognite delivered applications running in NOC infrastructure. NOC is responsible for the response related to and recovery of NOC managed infrastructure.

Cognite incident roles are defined in the Cognite incident process and BCP/DR documentation. These roles will also take effect in the event of a major incident affecting NOC.

- Incident Commander: Coordinates incident response and resolution.
- Communication Lead: Manages communication around the incident.
- Operations Lead: Manages technical response and remediation of the incident
    - Investigation
    - Remediation
- Subject matter experts: Provides subject matter expertise and are managed by incident commander
    - Security
    - Legal
    - Product
    - Support

Customers will be notified of all Sev A incidents affecting them. Security breaches will always be treated as a Sev A incident,

Security program related to "software, service, and product assurance". Provide details on the testing regarding how both the internal and external layers are tested to ensure that code does not have vulnerabilities;	Please see clarifications in content moved from CT8 to CT3
Provide details about what are the escalation channels available in case of failure of any of the components part of the solution (for both custom and product services). **Further clarifications** Escalation channels: specify contacts to interact with in case of issue	Initial contact point is Cognite support where there are internal escalation paths depending on the severity of the identified issue, the issue timing, and the issue type. For security related issues or queries the alias security@cognite.com is the main contact point outside support. For Scale-up phase, an incident management group with both Cognite and NOC is established. **Answer to further clarifications** Escalation path for security incidents (as will be defined in the Rules of Engagement document): • Senior security contact in Cognite Security Insight & Control team • Cognite Chief Security Officer
Please provide Cognite latest penetration reports for both infrastructure and applications (in scope for NOC services). **Further clarifications** Latest penetration report: share the actual report on the test activity	The 2020-H1 penetration testing of the CDF API and services was just completed and is currently being reviewed. Will be shared as soon as the review is completed. **Answer to further clarifications** As per meeting Cognite will share the most recent pentest report as soon as the internal review and QA has been completed. The April 2019 report is being submitted with the status of the findings, once the April 2020 report available this will be shared as well.
Provide details about the most recent security incident response plans testing and what are the findings that NOC should be aware of. Provide details about if the incident response plan meet the SLA requirements for incident notifications as per the NOC requirements.	Comprehensive (full BCP/DR) test was completed in January 2020. This test simulated an complete region (europe-west1) outage and the rebuilding of services in a new EU region. This test confirmed Cognite ability to bring environment and services back online. From this test nothing specific to report to NOC.
Provide details alerting solutions able to communicate security incidents events.	Given the sensitivity and governance around security incidents such will not be part of an automated notification system rather communication will be through the established incident process and handling of this. There is a separate incident process defined for security events.

3.1.2 Identity & Access Governance

Security - Data Owner Control

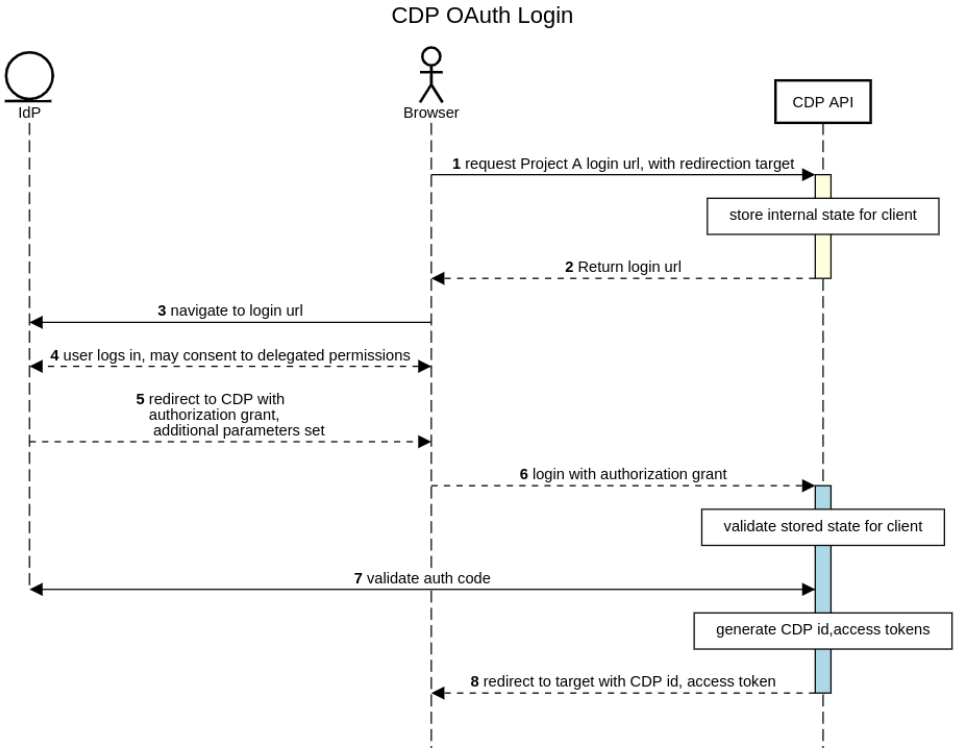
- Identity management and directory integration
    - Supporting OAuth2.0 and OIDC standards
- SAML not natively supported
- User access managed through data owner directory (for example Azure AD)

    - Enforce use of existing policies and procedures for identity onboarding/offboarding and granting of access
- Machine access via API keys
    - Managed by Data Owner through Console or API
- Information access management
    - Granular access control (R/W) to information objects, hierarchy, and resource types
    - Data Owner defines access classes and groups in CDF and own directory
    - Data Owner use own directory to manage access based on group memberships
    - 1-to-1 mapping between Customer Directory and Cognite CDF
- Reference to public documentation
    - https://docs.cognite.com/cdf/access/
    - https://docs.cognite.com/dev/guides/iam/

Security - User Authentication

- Users sign in to applications via CDF and the Identity Provider for the given project
- The application is issued an id token and an access token on behalf of the user
    - The id token provides information about the user
    - The access token is used to authenticate requests against CDF
- Applications use a variant of the OAuth2 implicit flow
    - This flow is a subset of an OAuth2 standard authorization code grant flow, with CDF as the backend and secret store

Security - User Authentication Flow



Security - Service Authentication

- Api keys: random secrets generated in CDF
- Associated with CDF service account
- Managed in CDF
- Attached as request header by service
- Use cases:
    - Daemon services
    - Long running jobs
    - Extractors
    - Scripts run locally

Clarifications	**Response**
Provide a list of all APIs available in the service including the distinction between which are standard and which are customized and detailing which are policies and procedures governing the use of APIs for interoperability between your service and third-party applications.	Please find updated documentation on the Cognite Documentation site: • https://docs.cognite.com/ • https://docs.cognite.com/dev/ • https://docs.cognite.com/api/v1/

Provide details about the eventual support of Multi-factor authentication options as per NOC requirement of hardware based, soft tokens or biometrics. (Note: any other form is not approved by the NOC.)	Multi-factor authentication is supported, defined, and enforced in the data owner (NOC) IdP (for example Azure AD). As a reference, Cognite default use is hardware security keys (Yubikey) and block use of weak multi-factor (example SMS).
Detail what is the security mechanism available to secure the root account.	For Google managed services this is handled by Google. For Cognite operated GKE workloads the best practices and up to date recommendation can be found in Hardening your cluster's security. This reference documentation is maintained by Google and also used as "input" to Cloud Security Command Center rules. With regards to privileged access to Google Cloud infrastructure it is role based and limited to infrastructure and security personnel.
Detail what are the mechanisms available to protect against unused accounts.	User access is managed through the data owner (NOC) IdP and hence NOC is in control of account activation and termination. Such is done to ensure that there is one single source and alignment with data owner policies, processes, and procedures.
Detail what are the mechanisms available to protect against misuse of guest accounts.	User access is managed through the data owner (NOC) IdP and hence NOC is in control of account permissions. Such is done to ensure that there is one single source and alignment with data owner policies, processes, and procedures. In this example, NOC policies, processes, and procedures for granting access to guest accounts will be the controlling and gating factor.
Detail what are the mechanisms available to secure the cloud service administration portal.	• Role based access control ensuring that only authorized personnel (default infrastructure and security) have privileged access • Configuration managed through code • Configuration and permissions monitored through Cloud Security Command Center • Logging and audit trail as part of Cloud Audit Logs

Provide details about the procedures available for deprovisioning or attributes changes for accounts.	• Cognite accounts accessing cloud infrastructure:
	  ○ Deprovisioning
	  ■ Linked to the master employee database and removed automatically when status is changed/account removed.
	  ○ Attributes change
	  ■ Managed through infrastructure as code and using code change (SDLC) pattern including peer review
	• Cognite accounts granted access to NOC data in

	CDF: ○ Deprovisioning ■ Managed by NOC through NOC directory ○ Attribute change ■ Managed by NOC through NOC directory and/or Cognite Console • NOC accounts granted access to NOC data in CDF: ○ Deprovisioning ■ Managed by NOC through NOC directory ○ Attribute change ■ Managed by NOC through NOC directory and/or Cognite Console

3.1.3 Infrastructure Security

See section 3.1

3.1.4 Application Security

See section 3.1

Clarifications	**Response**
Provide details about how CDF connects to GCP from an Infrastructure perspective (e.g., a dedicated or partner interconnect, etc.) and what are the POPs/edge connectivity for them.	CDF runs on top of Google Cloud Platform, for NOC the scope is EU and region europe-west1.

3.2 CORE TOPIC 9 - DATA SECURITY

3.2.1 Data Classification / Segregation

Information Classification

Classification of Information - Cognite

Classification criteria
The level of confidentiality is determined based on the following criteria:

- value of information – based on impacts assessed during risk assessment
- sensitivity and criticality of information – based on the highest risk calculated for each information item during risk assessment
- legal and contractual obligations

Information Categories
All information must be classified into Information Categories.

Information Category	Labeling	Description	Examples
Public	(unlabeled)	Information is not confidential and can be shared with no restrictions without any negative implications for Cognite. Loss of availability due to system downtime is an acceptable risk. Integrity is important but not vital.	• Product brochures widely distributed. • Information widely available in the public domain, including publicly available company web site areas. • Sample software source code. • Source code distributed under an Open Source license. • Financial reports required by regulatory authorities. • Newsletters for external transmission.

Internal	Internal	Information is restricted to management approved internal access and protected from general external access. Default access restriction to data is "ALLOW cogniters" Information can also be approved to be made available to identified external parties. Unauthorized access could influence Cognite's operational effectiveness, cause an important financial loss, provide a significant gain to a competitor, or cause a major drop in customer confidence. Information integrity is vital.	• Team discussions, presentations. • Meeting minutes. • Company directory. • Information on corporate security procedures. • Know-how used to process client information. • Standard Operating Procedures used in all parts of the Company's business. • All Company-developed software code not released under an open source license.
Restricted	Restricted	Available only to specified and/or relevant members, with appropriate authorization. Access is controlled via permissions (individuals or groups), default access is "DENY Cogniters" and "ALLOW group/individual". Sharing with 3rd parties is to be done via link sharing. Information should not be copied or	• This includes both personnel data and company (official) data. • Management Communication restricted to particular recipients. • Security Incident root cause analysis reports. • Breach Notifications. • Financial information. • Agreements. • GDPR Personal Data.

		shared as attachment. Sharing with external parties (individuals) requires signed confidentiality agreements. A breach of confidentiality could cause serious damage resulting in the compromise of activity within Cognite in the short to medium term.	
Confidential [Customer Data]	Confidential [Customer]	Information received from clients in any form unless otherwise specified by the data-owner/sender. The original copy of such information must not be changed in any way without written permission from the client. The highest possible levels of integrity, confidentiality, and restricted availability are vital.	• Client media. • Electronic transmissions from clients. • Customer data in Cognite Data Platform. • Information generated for the customer by Cognite production activities such as data that is calculated or derived from the original customer data.
Confidential	Confidential	Information collected and used by Cognite in the conduct of its business to employ people, and to manage all aspects of corporate finance. Information holds significant value for	• Information collected and used by Cognite in the conduct of its business to employ people, and to manage all aspects of corporate finance. • CDF Source code

		Cognite, unauthorized access can result in significant customer and/or financial damage. Access to this information is very restricted within Cognite. Strictly limited internal sharing to individuals only or special purpose groups. Number of individuals with access— especially with permanent access—is to be kept at a minimum. Access for external parties (individuals) requires signed confidentiality agreements and external approvals where applicable. Data is to be protected and encrypted while in transit and at rest. The highest possible levels of integrity, confidentiality, and restricted availability are vital.	

The basic rule is to use the lowest Information Category that ensures an appropriate level of protection, in order to avoid unnecessary protection costs.

List of Authorized Persons

Information classified as "Restricted" must have a List of Authorized Persons/Groups in which the information owner specifies the identities of persons/group who have the right to access that information.
The same rule applies to the Information Category "Company Internal" if people outside the organization will have access to such a document.

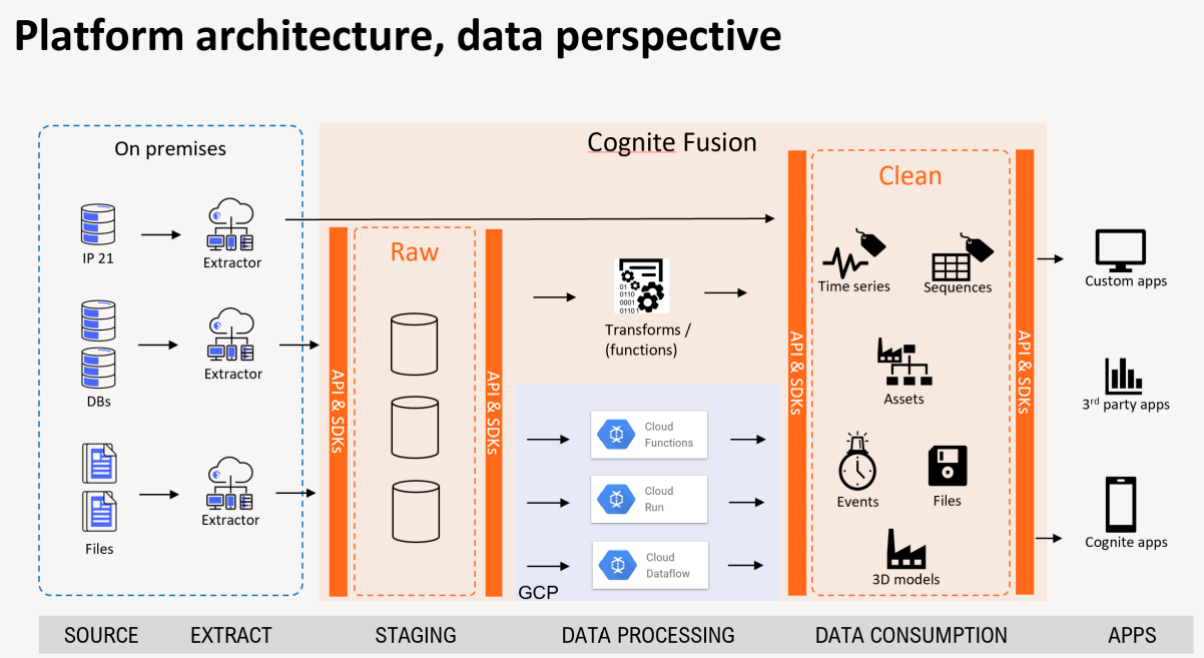
Classification of Information - NOC (Data-Owner)
NOC/Data-Owner can group and classify information using datasets for purposes of access and sharing. Management of datasets is through the Cognite CDF Console.

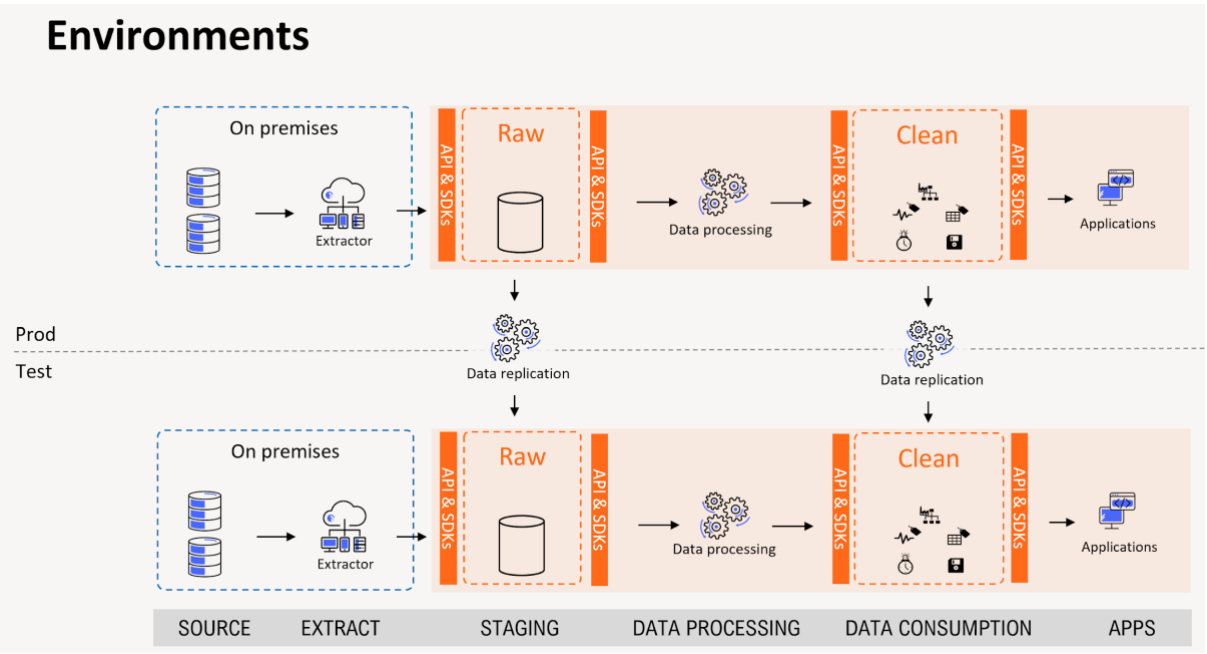
Clarifications	**Response**

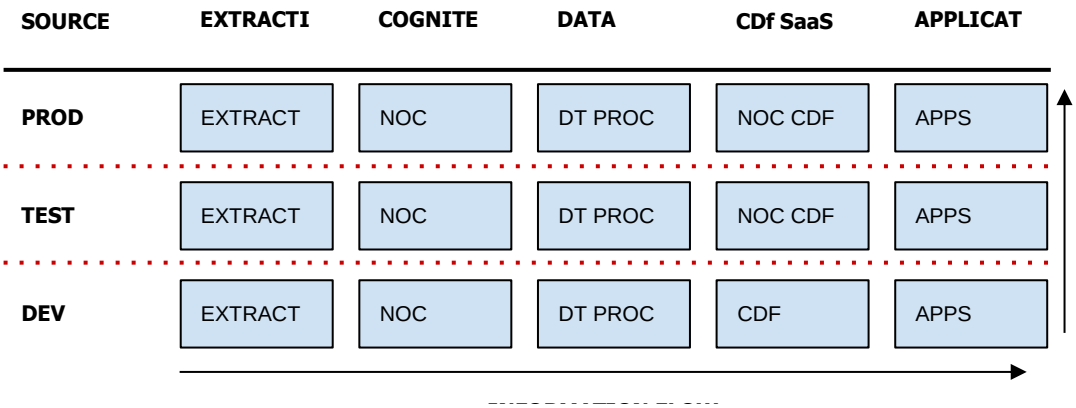
Information Classification. Provide details about how is data handled and/or protected and what are the protections that are been employed for data marked as Confidential [Customer Data] (NOC data). Define better what exacly "The highest possible levels of integrity, confidentiality, and restricted availability are vital" means in terms of data protection and integrity.	• Confidentiality
	  ○ Encrypted (at rest and in transit).
	  ○ RBAC/granular access control
	  ○ Not be generally available to Cognite organization, access to Cognite employees managed by data-owner.
	  ○ Cognite privileged accounts will have technical access. Privileged accounts mapped to role and need for access (default infrastructure and security).
	• Integrity
	  ○ Access control
	  ○ Monitoring and Audit trail
	  ○ Replication and backup
	  ○ Quality monitoring
	  ○ Testing and validation
	• Availability
	  ○ Built on top of robust and available services.
	  ○ Select and use technologies that support high availability (example stateful stores and container environment)
	  ○ Configure services and scaling to protect services against changing load and behaviour

Detail if there are any known data compatibility issues that NOC needs to be aware of thereby locking NOC data to Cognite services only.	No such lock-in exists, NOC can at any time access and extract the data through the Cognite APIs. In addition NOC remains the master data source.
Detail if it is possible to ensure that data does not migrate beyond a defined geographical residency.	This is the default behaviour, customer data is by default restricted to a geographical area. In the case of NOC this EU with active GCP region europe-west1 (St. Ghislain, Belgium).

3.2.2 Data Flows Inventory







3.2.3 Encrypted data transactions

Security - Encryption at Rest

- Based on provider (Google Cloud Platform - GCP) encryption by default
- Data is automatically encrypted prior to being written to disk using AES256
- Data for storage is split into chunks, and each chunk is encrypted with a unique data encryption key (DEK). These data encryption keys are stored with the data, encrypted with ("wrapped" by) key encryption keys (KEK) that are exclusively stored and used inside Google's central Key Management Service. Google's Key Management Service is redundant and globally distributed.
- Google uses a common cryptographic library, Tink, to implement encryption consistently across almost all Google Cloud Platform products. Because this common library is widely accessible, only a small team of cryptographers needs to properly implement and maintain this tightly controlled and reviewed code.

Security - Encryption in Transit

- When a user sends a request to a Cognite service, the information is secured in transit; providing authentication, integrity, and encryption, using HTTPS with a certificate from a web (public) certificate authority.
- Data the user sends to the Cognite service front-end is encrypted in transit with Transport Layer Security (TLS) Public certificate on Cognite gateway used to secure traffic aligned with NIST SP 800-52 Rev. 2
    - TLS 1.2 (or higher)
    - Strong TLS cipher suites (including)
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)

Clarifications	Response
Detail how encryption key are managed	Managed by Google, all services are using Supplier Managed Encryption Keys (encryption by default). Reference: https://cloud.google.com/security/encryption-at-rest/default-encryption
Define if is it possible for NOC to bring its own key to encrypt our data?	Not supported
Provide details about procedures in place to ensure production data not be replicated or used in non-production environments (e.g, data mocking, sensitive data randomization, etc.).	• Data ownership - Customer remains the owner of the data. • Policies governing use of customer data. • Data access controls where data owner controls access for Cognite employees
Provide details if there is any procedure which allows NOC to use its own key for encryption and use Azure Key Vault.	Not supported
Provide details about how the Secrets are managed (e.g., expiration date on Secrets, etc.)	Unclear what secrets it is referred to, for encryption keys please see the KEK and DEK details in the reference link. For API issued and used by the data owner such are managed by data owner (including rotation), this is administered through Cognite Console. Reference: https://cloud.google.com/security/encryption-at-rest/default-encryption

3.2.4 Data accessibility policies and procedures

Cognite staff that will have access to customer data will be in two categories.

A. Personnel granted data/information access via data owner established process and user directory
B. Personnel with privileged access to the underlying infrastructure.

For category A this number and type of access (read/write and data scope) will be directly controlled by the customer through issuing and revoking of user identities.

For category B
RO
Administrative access

Clarifications	Response
Referring to this section, please provide a more detailed description of the solution.	Default Cognite employees do not have access to customer data, such access is granted by data-owner through data owner directory. If a Cognite employee needs access to NOC data NOC need to a) create a user in the NOC directory and grant this identity data access or b) add the Cognite Azure AD as a guest user in the NOC directory and grant the guest identity access to data. For privileged identities/accounts (by default Cognite Infrastructure and Cognite Security), the privileged identity have technical access to data.

3.2.5 Data ownership & stewardship

Data sovereignty
Customer is and remains the owner of the data, as per the contract, Cognite is only a custodian of the data. Any details is part of, and defined, in the legal agreement between Customer and Cognite.

Clarifications	Response
Provide details about the responsibilities regarding data stewardship and how they are defined, assigned, documented, and communicated.	NOC is the data owner. Additional details around usage and data governance will be defined in the commercial agreement.

3.2.6 Data disposal

Data destruction
All customer information is securely stored, when and where applicable/necessary this information will be deleted/destroyed in accordance to the terms of the contract. Contract terms will honor and build on agreement with Google Cloud Platform.

Cognite will, at the end of a contract or on request from data owner, delete the tenant and all data connected to that tenant. Such delete will be completed within a period of 30 days after the initial request or contract termination. In addition, our cloud provider, Google Cloud Platform, has a long-term backup system to guard against natural disasters and catastrophic events. This system may preserve snapshot of systems for up to 6 months. These backups will not be directly available to Cognite, see details below.

Google Cloud Platform will enable Customer to delete Customer Data during the Term in a manner consistent with the functionality of the Services. If Customer uses the Services to delete any Customer Data during the Term and that Customer Data cannot be recovered by Customer, this use will constitute an instruction to Google to delete the relevant Customer Data from Google's systems in accordance with applicable law. Google will comply with this instruction as soon as reasonably practicable and within a maximum period of 180 days, unless EU or EU Member State law requires storage. For more info: https://cloud.google.com/terms/data-processing-terms#6-data-deletion

Clarifications	**Response**
Provide details about the following questions: • What are the data sanitization mechanisms (in details) provided to ensure that NOC data (including backups and archives) is securely deleted at the end of the contract? • Can NOC cryptographically make all its data useless in case we decide to terminate the contract? • Does Cognite have the capability to recover data only for NOC in the case of a failure or data loss?"	• All customer data is destroyed according to defined agreement or at the request of the data owner (NOC). Once the data destruction process has been initiated there is a delay before all data is fully removed from the Google systems (hot and cold). The initiation of this starts with Cognite deleting all NOC data, the rest is an automated process. • With the supported encryption key regime (Supplier Managed Encryption Key) such cryptographic revocation is not available. • Yes, if NOC information needs to be recovered (inside the terms of the SLA) Cognite will be able to restore such data and make it available.

4 FLEXIBILITY

4.1 CORE TOPIC 10 - PLATFORM CONFIGURABILITY

4.1.1 Parametrization

Parameterization offers adaptability and lets a system/module function well in different environments/changing operating conditions. Making a configuration change is cheaper than making a code change. As such it promotes reusability and increased value capture. This drives us to include and increase configurability.

At the same time, parameterization has a cost. It increases the code complexity as every parameter must be parsed and validated--it also dramatically increases the number of code paths and test conditions, and it represents a tail of maintenance and support cost.

We look at parameterization as a tool of great value that needs to be used with care. Every parameter needs to have a clearly identifiable value for it to be implemented.

This evaluation takes place centrally at Cognite for all our products and services.

For the project deliveries, the use case work streams is the arena where the appropriate parameters are identified. The use case requirements and operating conditions drive the decisions on what to parameterize and how to surface those controls.

Clarifications	**Response**

Detail the approach and the principles that could affect cross UCs points. The following could be some examples grouped by architectural layers:
• Infrastructure
  ○ Throttling (e.g., resources usage thresholds, etc.);
  ○ Infrastructural resource references (e.g., environments names, etc.);
  ○ Logging (e.g., e.g. logging verbosity level, log rotation mechanisms and frequency, etc.);
• Data
  ○ Data source references (e.g., data sources name, etc.);
  ○ Data replication mechanisms (e.g., configure
	• Infrastructure:
	  ○ Max usage boundaries (CPU and RAM)
	• Data applications (data pipelines + extractors):
	  ○ Connection credentials (source and target environments)
	  ○ Logging level
	  ○ Per module behavior (specific per UC per module), examples
	  ■ Data scope (i.e. run tests or debug runs on lower data volume).
	• Application logic
	  ○ Connection credentials
	  ○ Logging
	  ○ Most config is UC/module specific.

data replication across Cloud environments & data layers, etc.); ○ Data retention (e.g., configurable data retention for every stage of data storage raw and clean, etc.); ○ Data Quality checks configurations (e.g. tuning of data quality algorithms based on specific parameters or predefined sets of logic, alarms & thresholds); • Application logic ○ Assets configurations (e.g., platform, well and equipment addition or deletion, etc.); ○ Engineering inputs (e.g., thresholds, physical parameters, etc.); ○ Digital twin calculation logics (e.g., logics strictly related to physical appliances that could change, etc.); ○ Configuration of Users privileges on end users Dashboards (e.g. read-only, admin users with read+write privileges); ○ Enabling/Disabling tracing at environment level (i.e. not PROD, but only PRE_PROD and TEST); ○ Centralized configuration for all extractors including all available configurable items for extractors (e.g., logging, data source attributes, data source throttling, etc.); • Integration ○ External resources reference (e.g., URLs, etc.) ○ DevOps pipelines (e.g., CI/CD scripts configurability, etc.);	The list of parameters and configuration options is being discussed within use care work streams and can be found here: https://nocdrive.sharefile.com/home/shared/fo998105-ec2e-4350-8938-8822e6504821

4.1.2 Configuration externalization

The configuration parameters driven by the use cases are captured as a part of the use case work streams.

Some configuration parameters are use case independent. These include:

- Extractors. Extractors are configured via a local configuration file. The configuration file can be version controlled to track changes.
- Data pipelines (cloud infrastructure). Connection credentials are typically externalized to ease the promotion of pipelines/modules from one environment to another. Credentials secrets are stored in vaults (see 1.2.1.3.)

Clarifications	**Response**

Detail how it is possible to centralize all extractors configurations in the same configuration file **Further clarifications** Please detail what are the main processes that you propose in order to manage hot and cold configuration variables changes (e.g., configuration variables changed in a configuration file and application restart, variables changed in a custom developed dashboard and no application restart needed, etc.)	Different extractor types employ different configuration file schemas, so you should not share them across types. Also for extractor instances of the same type (ex. The db extractor), you usually have some uniqueness per instance. For example, the source query specification. This makes it harder to share configuration files. All configuration is based on files, so you could centralize the management of the config files by hosting them on network drives. **Answer to further clarifications** For extractors you need to restart them in order to pick up configuration updates. For Cloud Run modules/services, the main parameters are connection credentials (dev, test, prod, etc.), and these are injected at deployment time by the deployment pipeline. For the use cases, the current state is roughly as follows: • The UCs have chosen to reduce the number of configurable parameters • Most of the configurable inputs are surfaced in the dashboards themselves as user input fields/drop-downs
In addition to secrets, what are tools/technologies adopted to store and manage other configurations?	In general, the tools chosen depend on the hosting runtime of the module. We try to encapsulate the configuration/parameters at the source code level so that these can be pushed to the appropriate configuration mechanism. For example, CloudRun modules can read config parameters via injected environment variables in their container runtime.

4.1.3 NOC self-configurability

The specifics of configurability is covered by the use case work streams.

Clarifications	**Response**
How will NOC be able to access and change parameters? Refer to cross UCs parameters that you will specify in the section 4.1.1	The list of parameters and configuration options is being discussed within use care work streams and can be found here: https://nocdrive.sharefile.com/home/shared/fo998105-ec2e-4350-8938-8822e6504821 Once all the solutions are finalized, they will be documented accordingly.

Cognite AS
Fornebuporten, Oksenøyveien 10
1366 Lysaker, Norway

4.2 CORE TOPIC 11 - MONITORING AND ALERTS

4.2.1 Logging

4.2.1.1 On-Premises components logging

The on-premises components (the extractors) log to the local disk. The log location (filename and location) and log level is specified in the extractor configuration file. NOC is responsible for the on-prem extractors and has thus defined where these files should exist on the server.

If one want to get access to the logs outside of the extractor server(s), there are mainly two options:

1. Configure the extractor to export the logs in addition to log locally. Some extractors, like the db-extractor, has this as an option if NOC wants to use Stackdriver/Cloud Logging
2. Install an agent that exports the local logs outside of the server

Logging configuration

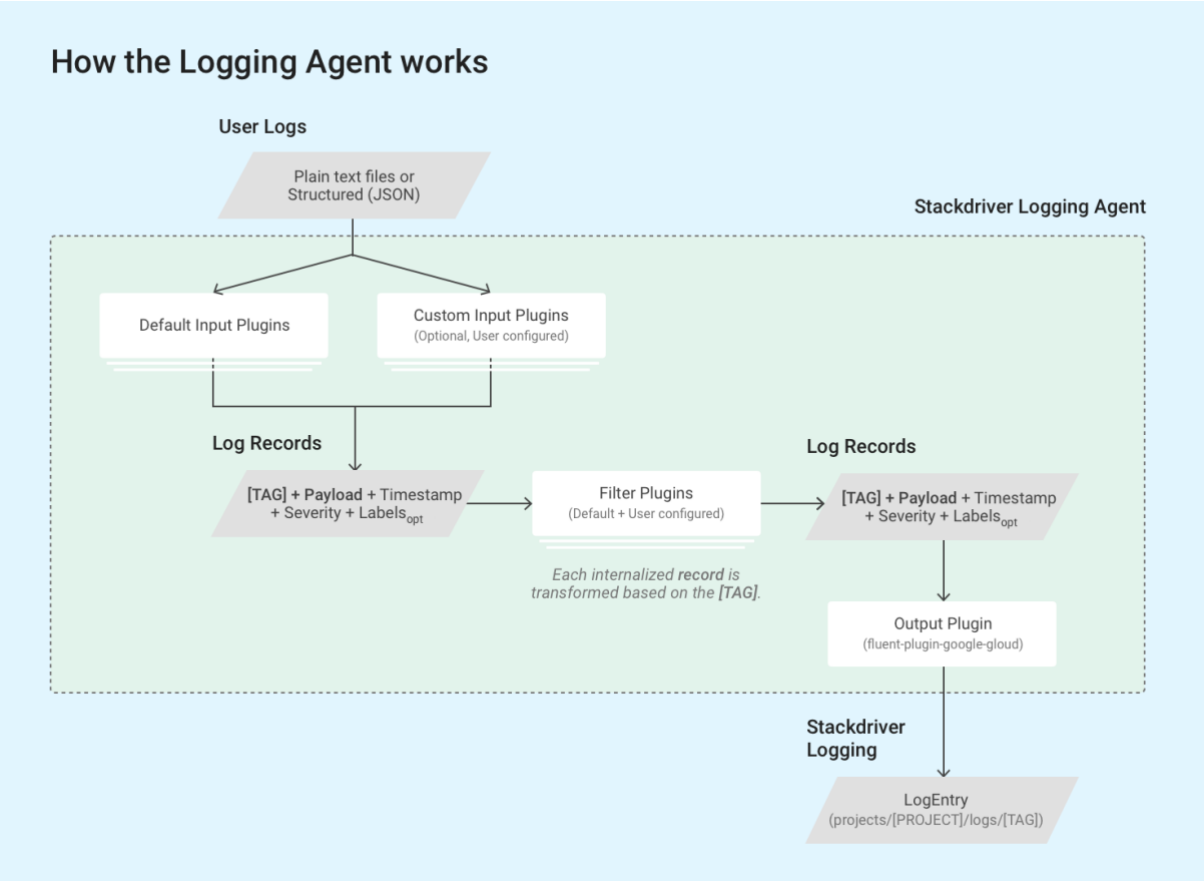
- Logging level
- Logging target (file or via agent)

Clarifications	**Response**
Provide details about Prometheus and Grafana on-premises instances in terms of configuration and deploy best practices in order to have a robust system that allows NOC to deeper monitor extractors execution. Sharing the acquired expertise and knowledge, Cognite should support NOC to setup and configure Grafana and Prometheus	Prometheus is a timeseries databases which can be set up as a target for all on-premises installations we provide. Grafana can be set up internally and configured to visualize such metrics. Both tools are open source and have rich extensive documentation available on the Internet. As the best practice we recommend to install a single instance of Prometheus and Grafana so NOC IS SMEs would be able to observe the same metrics as Cognite Support has internally. There is no need to set up more than a single instance since observability services downtime won't impact any of the extractors. Cognite can support NOC IS SMEs if there are any questions on how to set this up
Further clarifications Specify if a log rotation setting is provided by the extractors in order to ensure timely clean up and avoid storage filling	**Answers to further clarifications** Every extractor comes with a built in log rotation feature of 7 days. That can not be configured externally at the moment.

4.2.1.2 Cloud components logging

Project cloud components implement logging via standard logging libraries (ex. Slf4j + logback for Java) so that the log output can be easily configured for different hosting environments. In the Cognite operated environment, the components will primarily log to stdout/stderr which is picked up by the central logging system (Cloud Logging).

Cloud product logging is using native GCP logging (Cloud Logging) and native transport google-fluentd. The infrastructure is default/native GCP/GKE.



Audit logging is using a separate log service that is developed and maintained by Cognite. Audit logging engine is deployed as a separate sidecar in the GKE infrastructure (using standard deployment/operations pattern). The audit log sidecar sends data Cloud BigQuery for analysis and insight.

Access to logs is through GCP Cloud Console or related API (GCP native).

4.2.1.3 Centralized logging across all layers

Default logging system
All infrastructure is using/running on GKE and using GCP native Cloud Logging.
https://cloud.google.com/logging

UX/UI Cloud Logging
https://cloud.google.com/logging/docs/view/logs-viewer-interface

Accessibility logging console
Part of the Google Cloud Console with access is controlled via IAM, not accessible to customers/data-owners..

CDF user activity logging
Providing customers with restricted access to or custom export of CDF audit logs can be arranged per agreement with Cognite Security Team. There is work in progress on providing such access via an API.

There is no centralized logging system exposed to NOC. Cognite employs centralized logging as a part of product operations and project cloud infrastructure operations.

4.2.1.4 Logging levels and configurability across environments

Extractors' log levels are specified in the extractor documentation.

4.2.1.5 Logging Tools and other related features

Clarifications	Response
Specify which extractors allow to configure logging and detail what and how configurations can be modified to directly alter their behaviour (e.g., logging level, logging rotation, etc.)	This is specified in the product documentation accompanying each extractor.
Provide details about the possibilities to redirect all logs (i.e., Extractors and custom Cloud components) in an external centralized repository (i.e., Cloud or on-premises system)	**Extractors:** Logging configuration and destination system, based on the functionality described above, will be managed by the operator of the extractors. **Custom cloud components:** These are running on the Cognite SaaS infrastructure and the default logging will be to the native observability stack of the

	cloud environment (GCP at the time of this writing). Currently there is no functionality or plan to integrate/redirect logs to external systems.
Provide details about the possibility to use NOC local Prometheus and Grafana for extractors performance monitoring	This would be a supported and recommended pattern.
Detail what are functionalities provided by the CDF console in terms of monitoring and logs analysis	Focus in Cognite Console is on data monitoring including data quality. This should be part of the end-to-end monitoring for data flow (is data being received) and data quality monitoring. Alerting should be setup according to the operational pattern.

4.2.2 Metrics

4.2.2.1 On-Premises components metrics

On-premises components (extractors) emit Prometheus compatible metrics. Different metrics are available per extractor type. It is not possible to configure the metrics themselves, but your metric receiver (i.e. Prometheus) can be configured with custom rules.

For example, NOC could set up a Prometheus instance to receive metrics from the extractors. A Grafana dashboard could be used to visualize the metrics from Prometheus.

4.2.2.2 Cloud components metrics

Cloud components are monitored using Kubernetes Engine Monitoring
https://cloud.google.com/monitoring/kubernetes-engine/observing

Observing your GKE clusters
https://cloud.google.com/monitoring/kubernetes-engine/observing

Prometheus + Grafana. Hosted internally not exposed to customers/data-owners. Scraping every 15 seconds

4.2.2.3 Other Event Metrics

Clarifications	**Response**

Provide a detailed list of metrics currently in place to monitor all the components of the system	The full fidelity (detailed list) of observability/metrics will not be listed here but rather a set of examples ranging from product team level to aggregated (CDF) level. For product teams the metrics monitored and their alerting levels will greatly vary from low level disk and iops metrics (database/Bigtable) to end-user application experience (performance and visualization). For support and use-case monitoring tracing metrics is a key data source and alerting trigger. Furthermore there are metrics throughout the entire product life cycle, from source code (examples include vulnerability, build, test, perf.) through operating and running with metrics derived from distributed tracing.

4.2.3 Tracing

4.2.3.1 Tracing for On-Premises components

There are no specific tracing components offered by Cognite. In order to inspect the behavior of extractors, we recommend switching on more verbose logging. Of course, it is possible to enable deep Windows process tracing but that is up to your own discretion.

4.2.3.2 Tracing for Cloud Components

Tracing for cloud components operated by Cognite, both product and project, are not made available externally. Lightstep is used for tracing.

4.2.3.3 Tracing Tools Configuration for different environments

Tracing for components operated by Cognite (i.e. CDF and project cloud infrastructure) is not available for external configuration.

Clarifications	**Response**
Define what are the tracing capabilities in terms of functionalities provided by Lightstep (e.g., outlier identification, sampling tracing, etc.)	There are a few areas where Lightstep excel and these include: • No sampling, meaning Lightstep process all trace data without negative impact to (performance/cost). This is enabled through the satellite infrastructure. • Powerful detection and correlation engine with strong root cause analysis for large and complex service architectures.

	• Flexibility in trace processing and analysis through mature configuration and use of streams. • Ability to integrate with backend log infrastructure for improved troubleshooting, insight, and resolution time.

4.2.4 Alerting Mechanisms

4.2.4.1 Alerting Tools

The alerting and troubleshooting mechanisms are described in the chapter 1.2.3.

4.2.4.2 Issue troubleshooting process

The alerting and troubleshooting mechanisms are described in the chapter 1.2.3.

Quality Monitoring
When you rely on data to make operational decisions, it is critical that you know when the data is reliable and that end users know when they can rely on the data to make decisions. Quality monitoring is available in the Cognite Console.

Monitor
A data quality monitor is used to monitor a collection of time series, typically related to a particular use case.

For each monitor, you can specify a name and a description. You can also choose who else should have permission to edit the monitor. All other users have access to view the monitor.

Rule sets
Within each monitor, rule sets let you group time series with similar data quality requirements. You can specify quality requirements for a group of time series at once using rule sets.

Output from quality monitoring
The data quality monitor outputs events and time series to CDF.

These output events and time series let you report the data quality status in other apps and dashboards. They are also the basis for alerts via email or webhooks.

Events

If the data quality does not meet the requirements, the data quality monitor writes an event to Cognite Data Fusion (CDF).

The event is of the type Data Quality Monitoring Alert, and includes a sub-type corresponding to the type of rule that was broken

The metadata fields contain information about which monitor and rule set the event belongs to, and which time series broke the rule.

As the quality incident progresses, the event will be updated with the aggregated values (like the list of all time series that has been broken, the worst value for the whole incident etc.).

When the data quality is restored and meets the requirements, the event is updated with an end time, and the isOpen metadata field is set to false.

Time series
For every data quality rule, we generate a few output time series with metrics.

The three data quality metric time series are:

- **Broken time series count**: Count of broken time series for the rule. A high value means that many time series in the rule set are breaking the specified rule. Includes time series with abnormal errors, e.g. if there are zero data points in the time series or if the time series has been deleted.
- **Worst measurement**: This is the worst measurement within the evaluation window across all time series in the rule set.
- **Data quality score based on ratio of broken time series**: A score between 0 and 1 that estimates the current data quality for the rule. The calculation is (1 - num_timeseries_broken / num_timeseries_in_ruleset). A high value means higher data quality, as fewer time series are breaking their data quality requirements.

A data point is added to each of these time series every time a rule is evaluated

Alerts
You can set up alerting via email or a webhook URL when data quality is broken.

The alerts are sent whenever a data quality event is opened or closed.

- An event is opened when a rule in a rule set is broken. The event stays open as long as any time series in the rule set are breaking the rule, even if the quality of some time series is restored.
- The event is closed when no time series in the rule set break the rule, i.e. the quality is restored.

You can specify up to 5 email addresses to send alerts to

You can use a webhook to send notifications through other channels. A webhook lets an app provide other applications with real-time information. For example, you can use a tool like Opsgenie to receive the notification and pass it on to the relevant recipients through email, Slack, or other mediums

Clarifications	**Response**
Provide details about processes currently in place to provide visibility about failures on CDF side to NOC (e.g., extraction of logs from CDF central logging system in certain situations, access to tracing logs for specific cases, etc.)	Where applicable such information can be part of an incident report and/or postmortem.
Detail what are the procedures currently in place for alerting and what is the process in case of failures to notify NOC and react to the error	Handling of failures and incidents in Cognite follow the established process where one key activity is communication (including customer communication). Default such communication will flow through the NOC account team.
Define what is the current procedure in place for operational issues (e.g., low reactivity in dashboards, etc.) alongside with the technological stack and alerts system used to track and monitor them.	Issues, when detected and/or reported, will be logged and managed by the Cognite support team. Depending on the nature of the issue (including root cause and impact) the support team will work with NOC or internal Cognite team to resolve. The support team will track/monitor and keep involved parties informed of status. Tooling includes GCP observability stack (Cloud Monitoring & Logging), Lightstep, Opsgenie, and Jira.

5 STANDARDIZATION

5.1 CORE TOPIC 12 - AZURE RE-PLATFORMING

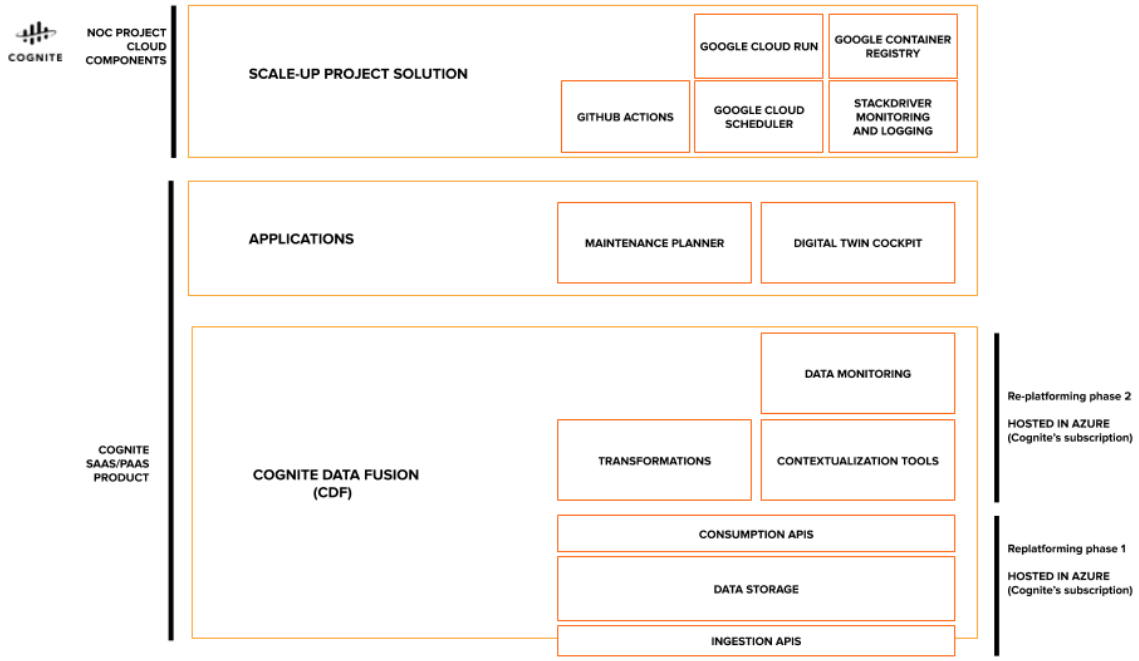
Currently, Cognite is re-platforming Cognite Data Fusion (CDF) to run on the Azure cloud platform. The re-platforming will make CDF run in Cognite's Azure subscription.

The current re-platforming does not include applications such as the Digital Twin Cockpit and Maintenance Planner currently deployed at NOC.

Nor does the re-platforming include solutions specific to the NOC scale-up project that depends on cloud components such as GitHub Actions or Google Cloud Run, Google Cloud Scheduler, Google Container Registry, Stackdriver etc. Re-platforming the project specific solutions is currently not planned for.

Cognite Data Fusion is a SaaS product that Cognite is responsible for operating and managing. This will remain the same after the re-platforming to Azure. CDF will run in a Cognite controlled Azure subscription. CDF exposes a set of APIs and graphical user interfaces that creates an abstraction level between the underlying cloud components in use, and the clients. Cognite will continue to evolve the SaaS product and use the most suitable storage and compute technologies available on the particular cloud platform, also in Azure.

The figure below illustrates which components are re-platformed to run in Azure.



5.1.1 Cloud components

5.1.1.1 Data Storage

Data will be stored in the Cognite Data Fusion SaaS product. CDF provides an API for storage and retrieval of data with various SDKs and connectors for data connectivity. Aspects like Performance, High Availability and Disaster Recovery for the storage layer is fully managed by Cognite in the CDF SaaS offering, and governed by the prevailing SLA.

CDF will persist data in Azure using Microsoft's disk offerings that provide appropriate performance. Cognite target the performance in Azure to be akin the performance of CDF on GCP. High availability and disaster recovery is enabled in Azure through persisting data redundantly across multiple disks and data centers. In addition to disks, backups will be stored in Azure managed services that provide high availability and disaster recovery.

Clarifications	**Response**
Specify the components to migrate on Azure, or already migrated, and detail how the migration process has been managed in order to ensure no impact on business continuity and requirements compliance	In phase 1 the data storage component will be replatformed. Separate environments can be set up using CDF's Azure hosted Data Storage APIs for validation purposes and to migrate applications. The environment set up will handle

	ingesting data ingestion into the new CDF instance in Azure. After setting up, a period of validation from NOC and Cognite should take place, and the final decision to migrate production environment from GCP based environment to Azure based environment will be jointly agreed when all criterias for business continuity are met In phase 2, Transformations, Contextualization tools and Data Monitoring components will be made available in CDF running on Azure. Separate environments can be set up using CDF's Azure hosted components for validation purposes and to migrate applications. The environment set up will add the data processing configurations to the new CDF instance in Azure

5.1.1.2 Data Processing

The data processing capabilities offered by the Transformations, Contextualization tools and Data Monitoring components of CDF is part of the SaaS offering, managed and operated by Cognite. These services will be enabled for Azure in phase 2 of our re-platforming efforts and will rely on Azure Kubernetes Services.

The availability of the data processing services will be built for the same requirements as for the CDF offering on GCP.

5.1.1.3 Applications

There are currently no plans for re-platforming applications to Azure.

5.1.1.4 Integration layer

The APIs of CDF are hosted within the SaaS offering and fully operated by Cognite. These are not based on GCP proprietary/managed services (on the contrary based on open source components) and will as per current plan be deployed and made available in Azure.

5.1.2 Connectivity

The APIs of CDF are exposed over HTTPS port 443 only accepting encrypted communication, this applies to extractors and applications. The re-platforming will not require any fundamental changes to this structure/architecture as will also be evident in the updated CAIQ for Azure (as per 5.1.3).

Clarifications	Response
Describe what could be the impacts on NOC infrastructure from a connectivity point of view	From the point of NOC the main change is in the physical distance between the source (NOC) and destination (from GCP in Europe to Azure in Qatar). From a logical perspective the the traffic change will be enabled through one of the following: • Traffic routing behind current API end-point • Change to the API URL • Change to the underlying IP The above enables a gradual and controlled change to the new underlying infrastructure All traffic is using the public Internet at the transport so there is no need to make any changes to the physical connectivity from NOC perspective, with the assumption that NOC does not have any specific rules in place handling traffic to/from the current CDF environment.

5.1.3 Security

We will use the same methodology and comparable technologies in Azure stack to achieve the same high level of security as on the Google Cloud Platform. The Flexibility and Observability will also be similar on the Azure platform.

The principals for the SDLC and security stack on Azure remain the same as is currently in place for GCP. As part of this there will also be an updated version of the CAIQ reflecting Azure. The availability of the CAIQ and the Cognite security stack on Azure will be fully aligned with the planned re-platforming - it is considered a gating requirement.

Clarifications	Response
Detail how Cognite is going to comply with security requirements currently in place when a component (e.g., service, application, DB, etc.) currently hosted in GCP will be migrated in Azure	From a principal point of view the same fundamental requirements will be applicable in Azure as currently in GCP. Security requirements are not tied to the cloud or architecture on the contrary the security requirements define and inform the architecture and implementation. Security requirements are defined through the following layers: • Foundational - Standards and best practices • Intermediate - Customer and industry verticals

	• Advanced - Subject matter experts and Cognite Security knowledge In practice this means that changes in technology/components will be measured against the Cognite Service Maturity Matrix (similar to BSIMM). This ensures a systematic and objective measurement of changes. This is not unique to the topic of re-platforming (GCP to Azure) rather a core way of how we operate and develop our products and services through continuous cycles.

5.2 CORE TOPIC 13 - MVP PLATFORM MIGRATION & RISK MITIGATION

5.2.1 On premise layers migration

The only software on premise as of MVP1 are the extractors. For the scale up, we will need new servers to run extractions into the new environments as explained in chapter 1.1.1.1 The configurations will be the same except for environment name and api-keys, unless we want to test a new configuration in one environment before merging this into the other environments.

5.2.2 Cloud layers migration

Most deliveries in the scale-up project are cloud based, either Cognite's products or other third party solutions.

5.2.2.1 Cognite Data Fusion (CDF)

This is where data resides along with our productised software

Data:
Data can either be fed with new extractors and then transformed in the new environments, or data can be replicated from another environment. As per discussions, we recommend the following guidelines:

One way data replication between environments is only done from MVP#1 PROD into

- DEV during scale-up development on a per-need basis
- TEST only when needed, before NOC sets up the necessary extractors into TEST
- See also chapter 1.1.2.

Transformations:
Transformations are made in the Cognite Console application. We also have an api and a cli that will make the queries available in GitHub and thus we can make tests etc before merging into the other environments.

Users and groups:
The intention is to use NOC IdP (AAD) as a central point to govern access management. For more granular permissions (specific data in CDF), Cognite Console will be applied.
The identity provider grants access to the users. Further, users are assigned groups within Console to provide different capabilities. The set of users are naturally different between

environments, when initiating a new environment one can copy groups and users from an existing one as a starting point.

Applications:
Cognite's own Software-as-a-Service applications (Console, Maintenance Planner) are available out-of-the-box when setting up a new environment. The Digital Twin Cockpit is a bespoke product for NOC built in the MVP#1 phase. As such it does require some minor manual configuration before being available in a new environment, which will be the responsibility of Cognite for new environments. Since it is hosted in our SaaS infrastructure, it is governed by our internal tools. The future of DT Cockpit is an ongoing discussion between NOC and Cognite will be addressed in other forums. Cognite's recommendation is to enter into a SaaS agreement, in the same way as for the Maintenance Planner application

5.2.2.2 Databricks

In MVP#1, the heavy transformation operations were running in Databricks. During the scale-up phase, we are moving this code into Google Cloud Run using GitHub, in order to ensure version control capability and automate testing, integration and deployment.

5.2.2.3 Visualization

In MVP#1, Grafana was the dashboard solution chosen by NOC and Cognite.
Currently, there are no plans to migrate these dashboards.

The deployment strategy Power BI is being discussed at the moment of writing this document.

For Plotly Dash dashboards, we recommend deploying it as a container in Google Cloud Run.

Clarifications	**Response**
Detail the suggested approach to migrate from the actual MVP#1 PRODUCTION environment to the new scaled-up PRODUCTION environment, covering both Application and data layer migrations, ensuring no unavailability of exposed services	The existing production environment will evolve and gradually integrate the functionalities from the scale up project. The promotion of any artifact (or a change in the data extraction) will be thoroughly tested by Cognite and NOC on close to production data in the test environment before moving to production itself. We will also monitor production and be ready to react if there are any problems. Ideally, we want to control this by having small changes integrated frequently. No changes to production will happen unless NOC decides and approves the change.

Referring to the paragraph "5.2.2.1 Cognite Data Fusion (CDF)" - Applications, clarify the sentence "The future of DT Cockpit is an ongoing discussion between NOC and Cognite will be addressed in other forums". Please detail the alternatives.	Cognite is committed to working with NOC to correct the imbalance between industry-grade UC2/ UC3 and MVP-grade UC1 ● The Digital Twin Cockpit was created as Use Case 1 of MVP#1. ● The Digital Twin Cockpit should be the "single source of truth" and host all use cases developed across the Digital Twin for all domains (Maintenance/ PO/ HSE etc). ● Support, Maintenance and License for the application was included through June 2020 ● Development of Use Case 1 was not continued with the Scale up project, due to oil price crash and COVID-19 crisis. ● The DT Cockpit has experienced challenges due to trade-offs made in the MVP#1 phase. ● The use of the DT Cockpit has been governed by the MVP#1 support and maintenance agreement that ends June 2020. Cognite suggest migrating to new SaaS product, that incorporates NOC's requirements for a robust, industry grade cockpit application ● Single interface for all current and future Digital Twin applications and use cases ● Single source of truth ● Single place to manage access control ● Filter access to applications and use cases / dashboards based on role and discipline ● All functional requirements from MVP#1 Digital Twin Cockpit captured ● Software as a Service licensing model, ensuring a future proof solution that improves over time ● To be released January 2021 A commercial proposal will be shared with NOC during June 2020

6 INNOVATIVENESS

6.1 CORE TOPIC 14 - MACHINE LEARNING & ADVANCED ANALYTICS READINESS

6.1.1 Advanced analytics features

6.1.1.1 Key data and analytics building blocks

Cognite Data Fusion is made to enable advanced analytics at scale through providing easy access to live streaming data, the ability to discover data, easy scaling of models across equipment, and through making it easy to move models to production, collecting feedback and yielding notifications. In terms of specific components:

- Data discovery, data access, and solution scaling are provided through the CDF API with several UI interfaces such as Asset Data Insight, Grafana, and the OData connector for PowerBI and Tableau for data discovery and the Console app for setting up ETL jobs and putting models into production.

- Putting Data Preparation into production and keeping track of lineage and Data Quality is provided through the Jetfire SparkSQL service as well as Cognite Functions both with UI through the Console app.

- Cognite does not offer analytics modelling capabilities or advanced preprocessing, but interfaces painlessly with best-of-breed tools such as Datarobot, H20, Azure MLOps, Google AI platform, and so on.

- Through the AIR application, Cognite offers Data Scientists a way to provide analytics functionality to SMEs, setting up notifications and collecting feedback. Refer to chapter 6.1.3 for more information on AIR

6.1.1.2 Advanced analytics out-of-the-box features

There exists a variety of constantly evolving best-of-breed tools for various kinds of analytics and the steps in such use case life cycles. CDF is a powerful and flexible enabler for these tools, offering complete freedom and ease of use in leveraging these tools to perform descriptive, predictive, and prescriptive analytics projects.

Some descriptive analytics functionality for end users is included as part of the CDF SaaS package with tools like Asset Data Insight and Data Studio in Console, whereby end users

can discover, investigate, and act upon patterns in historical data. Also, the Cognite application Automatic Identification and Reporting (AIR) is our way of building scalable data science in heavy asset industries. See chapter 6.1.3 for more description of its functionalities

Predictive and prescriptive analytics use cases are highly use case specific, thus require considerable customization. CDF's open APIs, SDKs and connectors to third party tools makes developing and deploying such advanced analytics projects faster, easier, and more secure.

6.1.1.3 Advanced analytics custom functionalities

Basically all DS use cases are solved in a notebook first. We have a solution called DSHub which offers Jupyter notebooks as a service for data scientists / data analysts development. It is configured with shared volumes (for notebook sharing), and comes with pre-installed CDF integrations and you can easily use the Cognite Python SDK for data exploration, feature engineering and afterwards you can start your model creation. Tutorials and templates are available with common tasks, so it is easy to get started. This is an environment where people can run Python, go through tutorials/templates, without having to worry about packages.

Cognite also offers a workflow orchestration service tailored for handling data science and ETL pipelines. A workflow is a group of actions, jobs or functions executed in sequence. The service includes:

- Possibility to schedule full workflows. Enabling the user to choose, transform, contextualize and model data in one intuitive sequence.
- Easy to understand UIs for creating multiple workflows and providing an overview of all existing data pipelines in your project
- Data Quality framework
- Retries, logging, metric collection and alerting integrated in the same service
- Possibility to share and reuse models and functions

This service, in combination with Cognite Functions, gives the data scientist the possibility to work in your well known notebook interface to create, train and deploy models. The workflow orchestration service gives an intuitive UI for combining various models in sequence and schedule at your convenience. Included is also data quality monitoring and status alerting for all jobs in the workflow.

As for the deployment process, we offer a solution called Cognite Functions which is a service that allows users to productize their code, which can be executed from an endpoint on demand, or be triggered by an event such as schedules. You can deploy your model in a fast and secure way by wrapping it up in a body function and from here applications and visualisation tools can call the endpoint to use the model output. For the future, we are

looking into a solution where you can store your functions and that afterwards can be picked and reused by other Data Scientists to solve use cases.

6.1.2 Distributed computing capabilities

6.1.2.1 Deep data advanced analytics architecture

Cognite Data Fusion provides a batch processing tool built on Spark and Cognite Functions, a function as a service (FaaS) tool. These two services allow users to explore and experiment with data through UIs and deploy code to a managed infrastructure through APIs and SDKs.
These processing tools can be scheduled and managed through both APIs and UI with observability and end user configuration.

6.1.2.2 Real time advanced analytics architecture

Analytics applications can use the CDF Spark data source to do batch or micro-batch (streaming) computations on data in Cognite Data Fusion in a distributed fashion. Queries to CDF support partitioning so they can be run in parallel on distributed compute frameworks. Queries also support change data capture via the last updated timestamp filter to do incremental loads.

6.1.3 Analytics solutions delivery - feedback and notifications

Automatic Identification and Reporting (AIR) Application
The Automatic Identification and Reporting application is Cognite's answer for scaling data science solutions in asset heavy industries. Large scale data science is made by professional data scientists (PDS) and consumed by subject matter experts (SME). The initial digitalization initiatives of Cognite with partners and customers made clear that both actors need to work together to make data science in the heavy asset industry a success. However, the challenge remains how to scale these successful stories. AIR provides a thin layer on top of CDF and other Cognite services that embraces both the PDS and the SME. AIR is providing the solution to the following problems in the large scale data science space.

Scheduling and orchestration of models is a daunting task when it comes to scaling data science. AIR is bringing data science closer to the software development lifecycle to deploy often and have a quick recovery rate. For this, data scientists are working in code versioning systems like Github. Pipelines take care of the deployment and AIR provides a staging and production environment to run integration tests before deploying to production and have

code reviews ensuring the quality of the deployed code. Additionally, this will increase the maintainability and improvability of the deployed models. Large scale data science solutions in the heavy asset industry are hardly one-off jobs but will need iterations and improvements as any software solution does.

Additionally AIR makes the person responsible for scaling that knows the equipment best:, the SME. Through the front end the SME can set up monitoring of sensors and assets of choice as well as configure their notifications settings.

AIR also provides historic backfilling out of the box. This is necessary for models that depend on past detection (comparing current behavior to past behavior) and for the data scientist to assess the performance of the model itself.

Finally, Labeled data is the essence of high performance machine learning algorithms. Unfortunately, the process of labeling data in the heavy asset industry is a tiresome process. AIR is giving SME the possibility of providing feedback on data science detection when they see it, instead of having them labeling a high number of occurrences. From a data scientist's perspective, a handful of high quality labels are preferred over a large number of low quality labels.

In sum, AIR not only provides the tools to make scalable data science but also provides an interface between the SME and data scientists that makes successful data science solutions possible.

6.1.4 Third party analytics engine integration

CDF has fully open, publicly available APIs, open source SDKs in several languages, as well as rigorous and stratified identity access management. This makes it easy to provide third party data modelling engines and application developers with access to historical and real-time NOC data hosted in CDF, and enables rapid development of tools to utilise this data. Resulting analytics output can easily be written back to CDF for consumption internally in NOC or by other third parties. A single cloud-native access point means read and write is virtually infinitely scalable, consistent with cutting edge development practices, and simplifies security and operation of third party analytics engines.