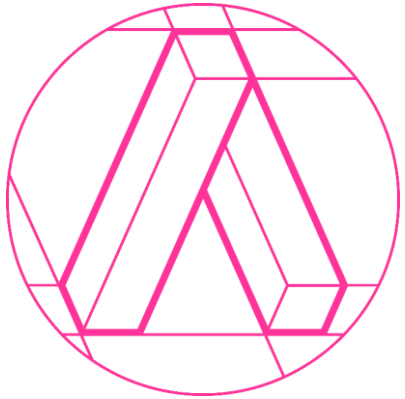


ARCAN

The solution for software architecture quality assessment

What is technical debt?

*«The implied **cost** of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.»*
[Ward Cunningham, 1992]

An example

Upcoming software release, client pushing

Developers are under pressure

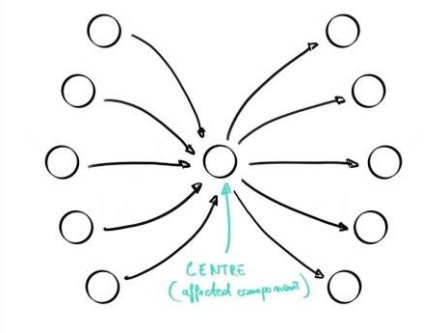
TECHNICAL DEBT GENERATION:

Fast solution, but not strategic
→ Gains time now, will be problematic in the future
→ Less efficient, **harder to secure**
→ Hard to modify, hard to evolve
→ **higher maintenance costs**

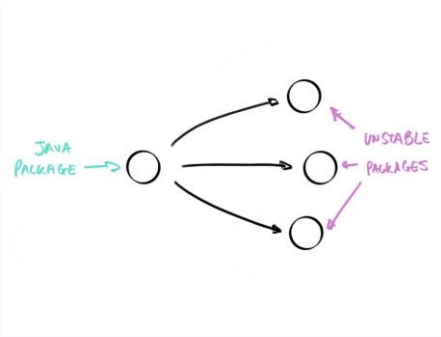
What are architectural smells?

Arcan detects *architectural smells*, software design decisions which negatively impact the maintainaibility, capability of evolve and security of the project. They are a symptom of Technical Debt.

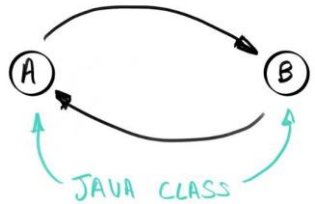
Hub-Like Dependency (HL): when an architectural component has (outgoing and ingoing) dependencies with a large number of other components. The component affected by HL centralizes logic, is a unique point of failure and favors change ripple effects. (detected on classes and packages)



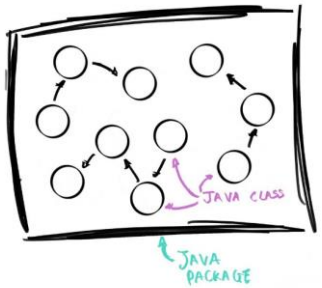
Unstable Dependency (UD): describes an architectural component that depends on unstable (prone to change) components. Instability is measured using R.C. Martin's formula. This smell can cause a ripple effect of changes in the system. (detected on packages)



God Component (GC): this smell occurs when an architectural component is excessively large in terms of LOC (Lines Of Code). God components centralize logic and are a sign of low cohesion within the affected component, increasing complexity. (detected on classes and packages)



Cyclic Dependency (CD): when two or more architectural components are involved in a chain of relationships. The parts of code affected by CD are hard to release, to maintain and to reuse in isolation. (detected on packages)



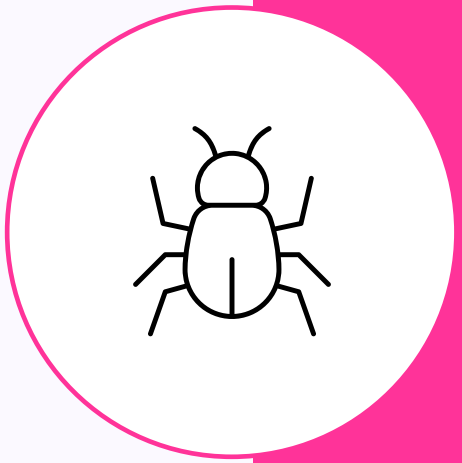
Renew the license

Contact info@arcan.tech to renew your license. Our team will provide a new license file.

Issues and improvements

If you find a bug or want to suggest a feature

Create a new issue on Github: https://github.com/Arcan-Tech/arcan-issues-public



ARCAN USER MANUAL

Glossary

- **Project**: the **software project** that you want to analyse with Arcan. It can be a folder containing source code or a Git repository (local or remote, e.g., hosted on Github).
- **Version**: Every time you change something in your project, Arcan tracks a new **version.** You can run Arcan multiple times on the different versions of the same project. Versions can be mapped to **commits** if you run Arcan on a Git repository**.**
- **Analysis**: a single **execution** of Arcan. An analysis is associated to a specific project's version.
- **Page**: a page of Arcan which contains **dashboards** or results
- **Architectural smell**: an architectural **problem** affecting a part of code. See slide #4 for more information.
- **Plot**: a **graphics** showing an insight or result
- **Dependency Graph**: the high-level representation of the project's **architecture**.
- **Container and Unit**: container=Java Package/ C++ folder; Unit=Java Class/C++ file

My projects

ADD NEW

View existing projects

You don't have any projects analyzed yet

ADD NEW

Analyse a new project

CREATE A NEW PROJECT

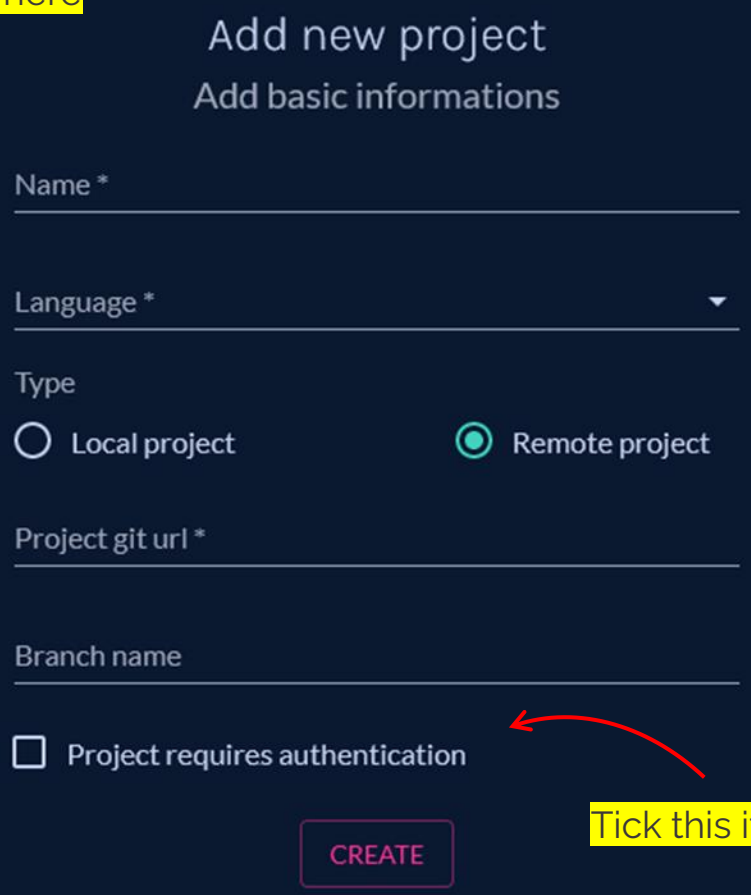
New project

CANCEL

Enter the name to be associated with the project here

The programming language of the project

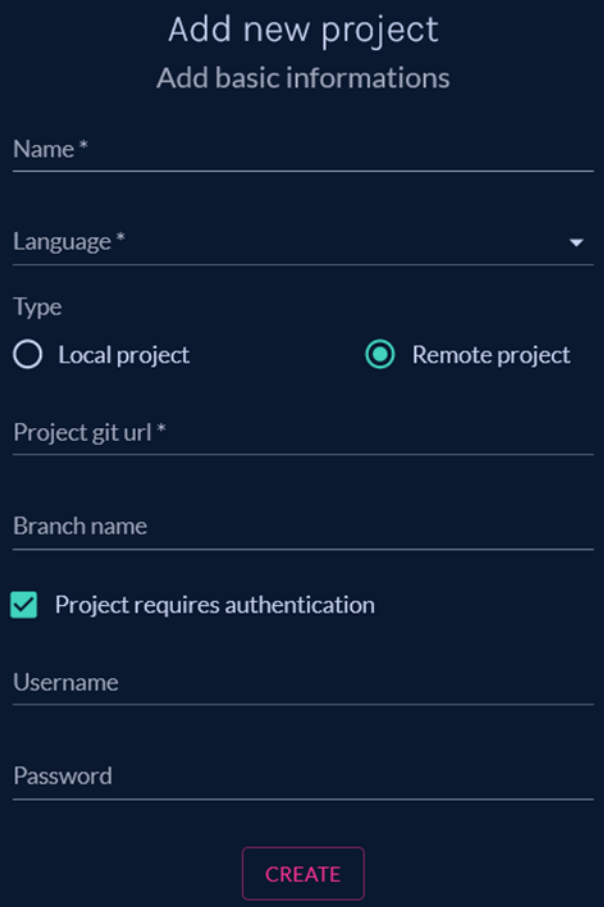
Choose here the type of Project to be analysed: **Remote** residing in a remote repository OR **Local** (w.r.t to Arcan server's location)



Tick this if accessing the project requires authentication

New project

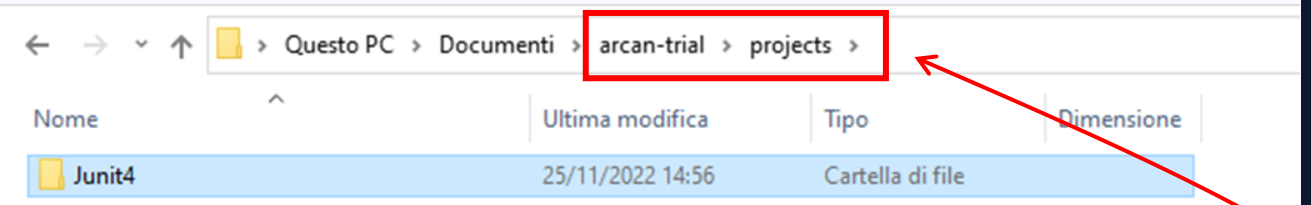
CANCEL



In the case of analysis of a remote repository requiring authentication, you need to create an **access token** on your repository provider. How to do that depends on the provider you are using. Here some instructions for the most popular ones:
- Github
- Gitlab
- Bitbucket:

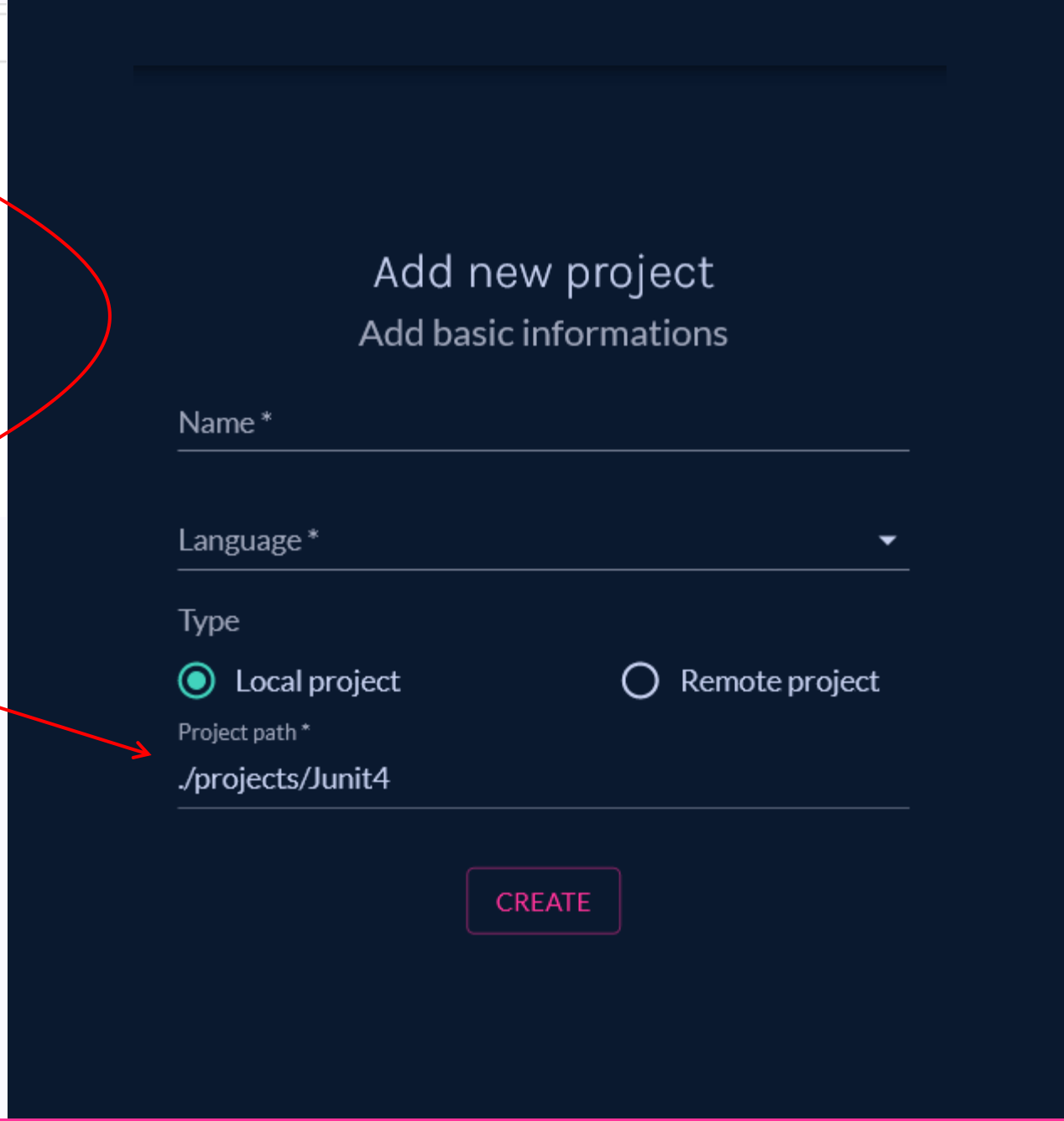
Insert here the username you use on the repository provider platform

Insert here the **access token** you generated

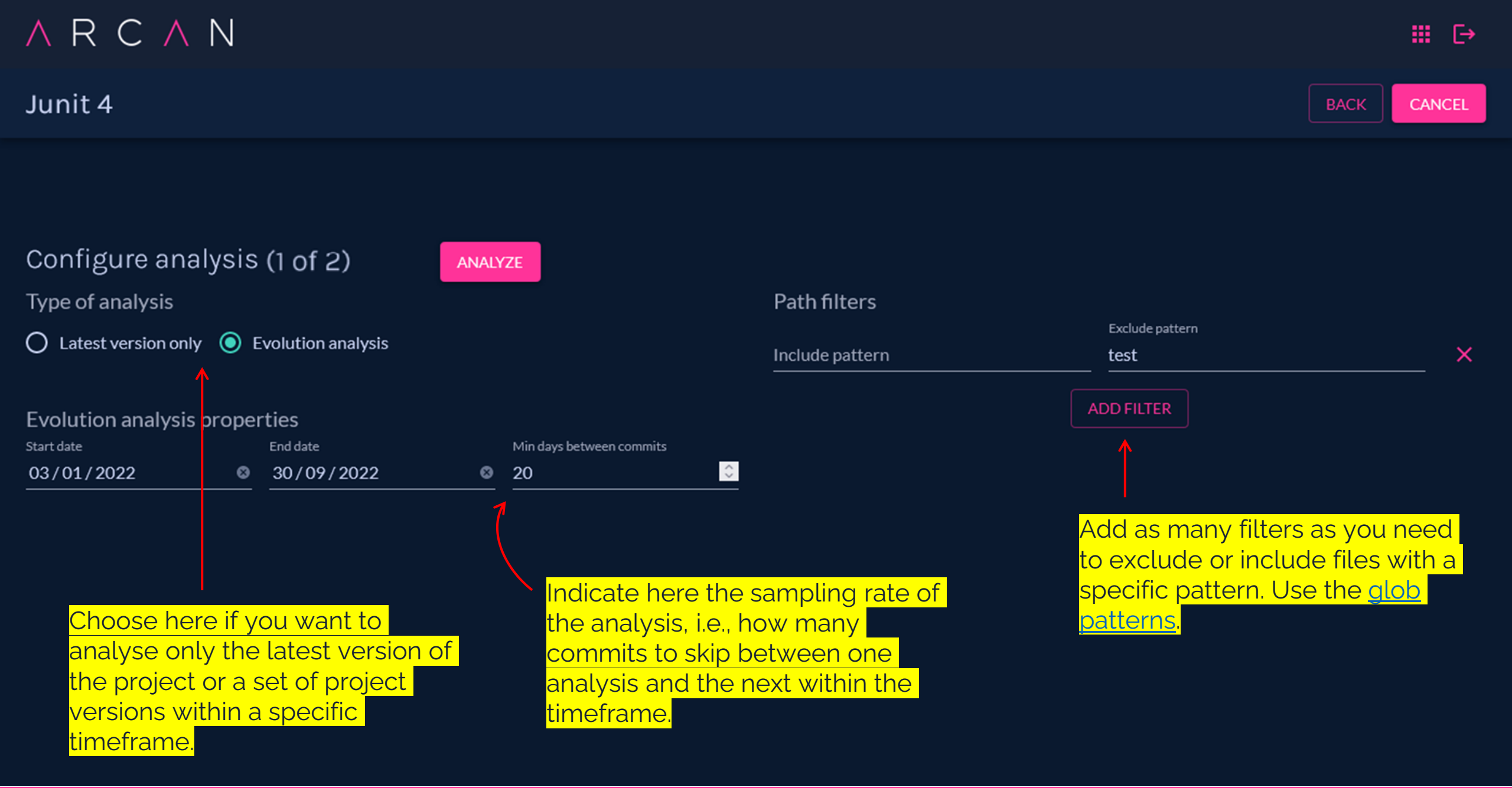


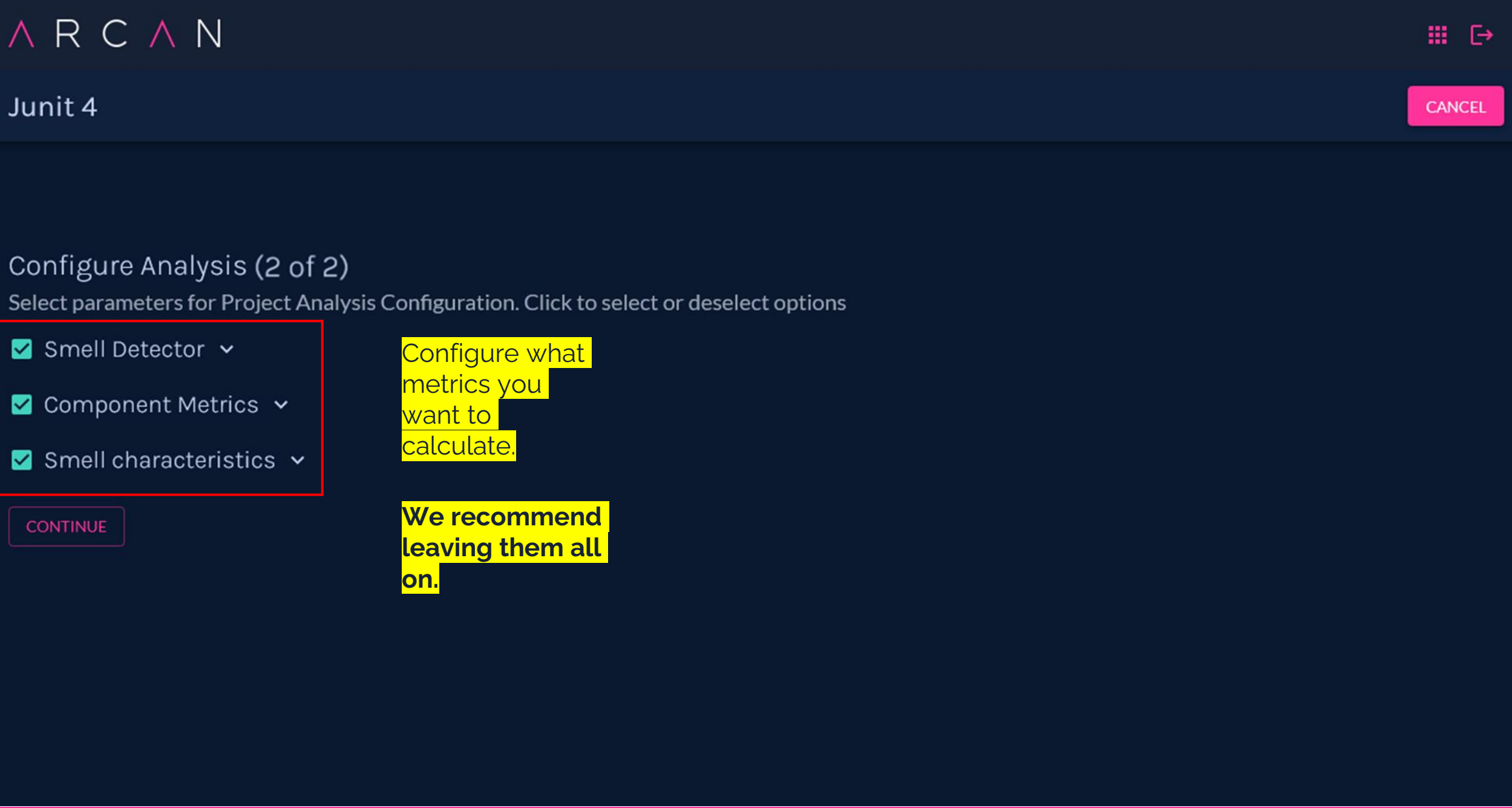
In the case of analysis of a local folder, you need to:

1) Copy the folder you want to analyse into **/arcan-trial/projects** folder.
2) In the Add New Project form, you must insert **`./projects/<folder_name>`**

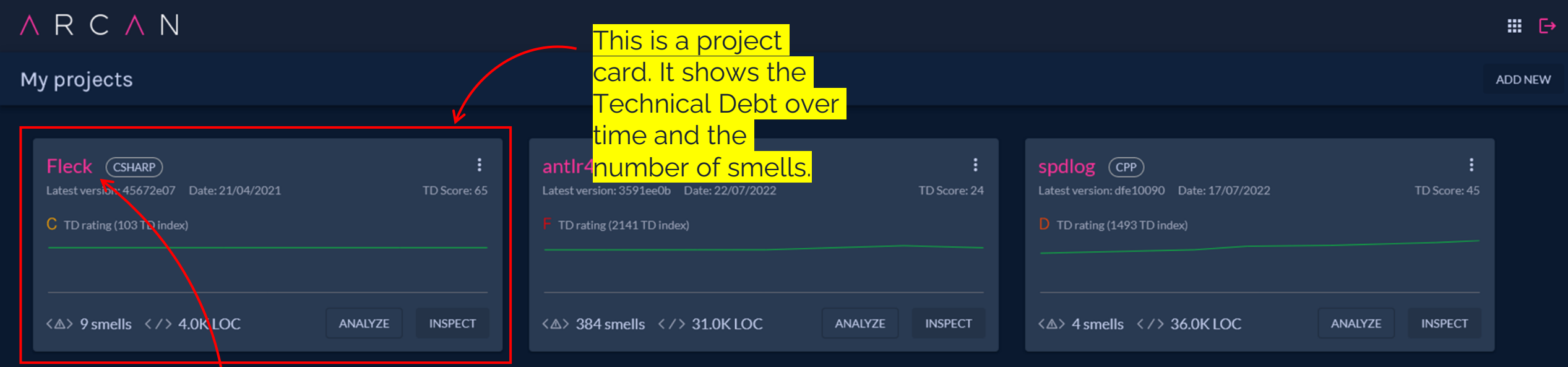


ANALYSE A PROJECT


ARCAN
Junit 4
BACK
CANCEL
Configure analysis (1 of 2)
ANALYZE
Type of analysis
Latest version only
Evolution analysis
Evolution analysis properties
Start date
03 / 01 / 2022
End date
30 / 09 / 2022
Min days between commits
20
Path filters
Include pattern
Exclude pattern
test
ADD FILTER
Choose here if you want to analyse only the latest version of the project or a set of project versions within a specific timeframe.
Indicate here the sampling rate of the analysis, i.e., how many commits to skip between one analysis and the next within the timeframe.
Add as many filters as you need to exclude or include files with a specific pattern. Use the glob patterns.


ARCAN
Junit 4
CANCEL
Configure Analysis (2 of 2)
Select parameters for Project Analysis Configuration. Click to select or deselect options
Smell Detector
Component Metrics
Smell characteristics
CONTINUE
Configure what metrics you want to calculate.
We recommend leaving them all on.

INSPECT A PROJECT



This is a project card. It shows the Technical Debt over time and the number of smells.

Click on a Project's name to inspect the results

TD index: the non-normalized amount of the TD generated by the architectural smells instances detected in the project. The higher the value, the more TD.

TD density: The TD Index divided by the total lines of code analyzed by Arcan (i.e. normalization). This value is used to calculate the TD score.

TD score: the percentile of the TD density with respect to a benchmark of other systems analysed by Arcan. The higher the value, the lower the TD density, and the better is the quality of the system. Example: 75 TD score = system has lower TD density than 75% of the systems in the benchmark.

TD rating: categorizes the TD score ranges into ranked categories ranging from A to F, with F being the worst possible rate (high accumulation of technical debt).

Junit 4
8774-3591ee0b - 22/7/2022

By clicking on the drop down menu under the project name, you can access the historical analysis, indexed by commit hash

Technical Debt Dispersion

Highest technical debt concentration:
org.junit.runner

Legend

- High concentration
- Avarage dispersion
- High dispersion

Technical Debt Spread

30%

-3.2%

Technical Debt

Hover the tooltip to show the explanation for the metric.

21.4
hours

-4.5%

Design Complexity

78%

0% =

Rigidity

Technical Debt Evolution

ARCAN

Junit 4
8774-3591ee0b - 22/7/2022

OVERVIEW ASSESSMENT COMPREHENSION

Navigate to other sections

Technical Debt Dispersion

Highest technical debt concentration:
org.junit.runner

Legend

- High concentration
- Avarage dispersion
- High dispersion

Technical Debt Spread

30%

-3.2%

Technical Debt

21.4
hours

-4.5%

Design Complexity

78%

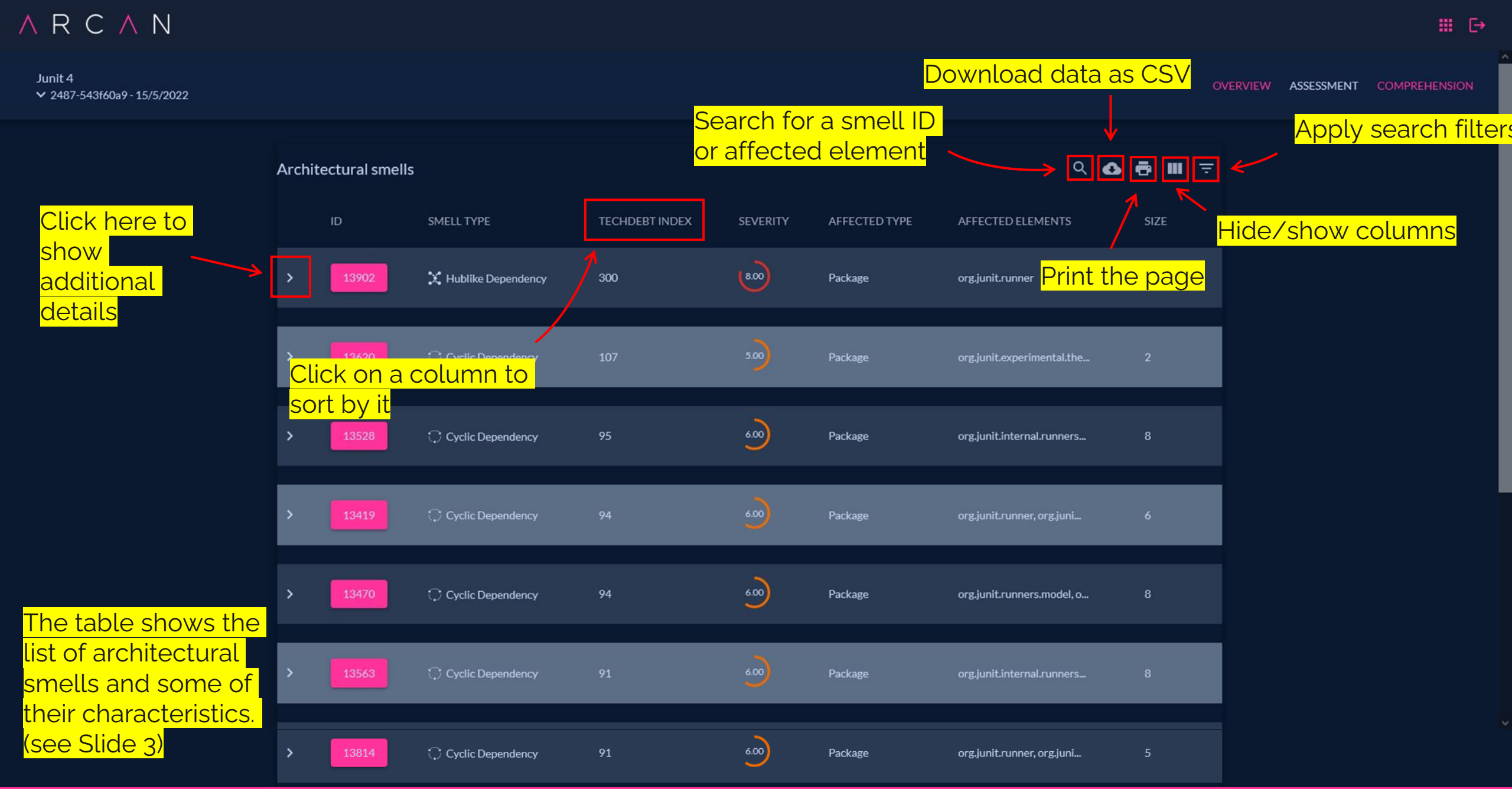
0%

Rigidity

Technical Debt Evolution

PAGE: **Assessment**

INSPECT A PROJECT


ARCAN
Junit 4
2487-543f60a9 - 15/5/2022
OVERVIEW
ASSESSMENT
COMPREHENSION
Download data as CSV
Search for a smell ID or affected element
Apply search filters
Hide/show columns
Print the page
Click here to show additional details
Click on a column to sort by it
The table shows the list of architectural smells and some of their characteristics. (see Slide 3)
Architectural smells
ID
SMELL TYPE
TECHDEBT INDEX
SEVERITY
AFFECTED TYPE
AFFECTED ELEMENTS
SIZE
13902 Hublike Dependency 300 8.00 Package org.junit.runner
13620 Cyclic Dependency 107 5.00 Package org.junit.experimental.the... 2
13528 Cyclic Dependency 95 6.00 Package org.junit.internal.runners... 8
13419 Cyclic Dependency 94 6.00 Package org.junit.runner, org.juni... 6
13470 Cyclic Dependency 94 6.00 Package org.junit.runners.model, o... 8
13563 Cyclic Dependency 91 6.00 Package org.junit.internal.runners... 8
13814 Cyclic Dependency 91 6.00 Package org.junit.runner, org.juni... 5

Click here to show the graph representation of the architectural smell

ID	Smell Type	TechDebt Index	Severity	Affected Type	Affected Elements	Size
13902	Hublike Dependency	300	8.00	Package	org.junit.runner	21

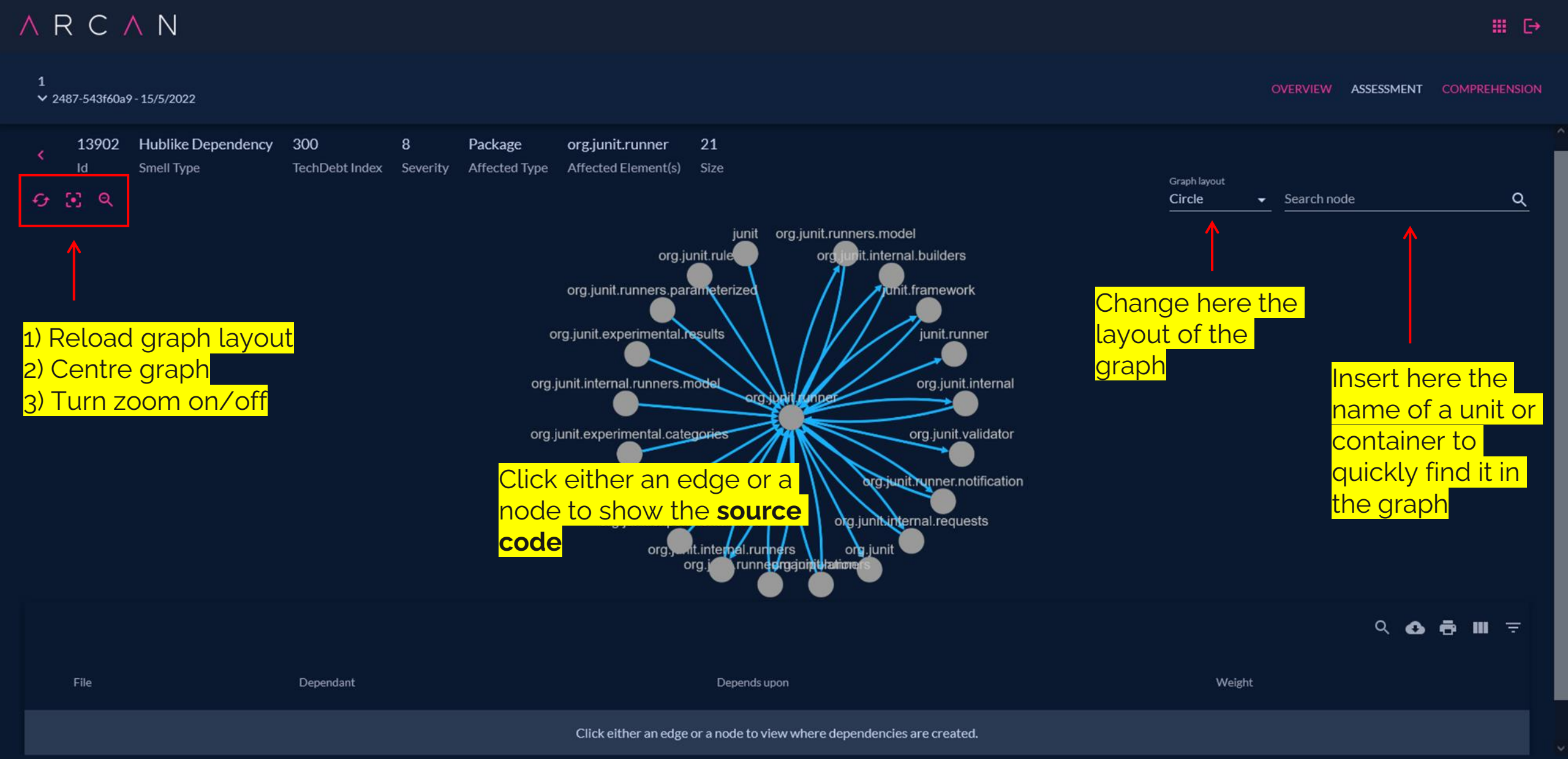
org.junit.runner
Affected Elements (complete)

17
Incoming dependencies (Fan In)

12
Outgoing Dependencies (Fan Out)

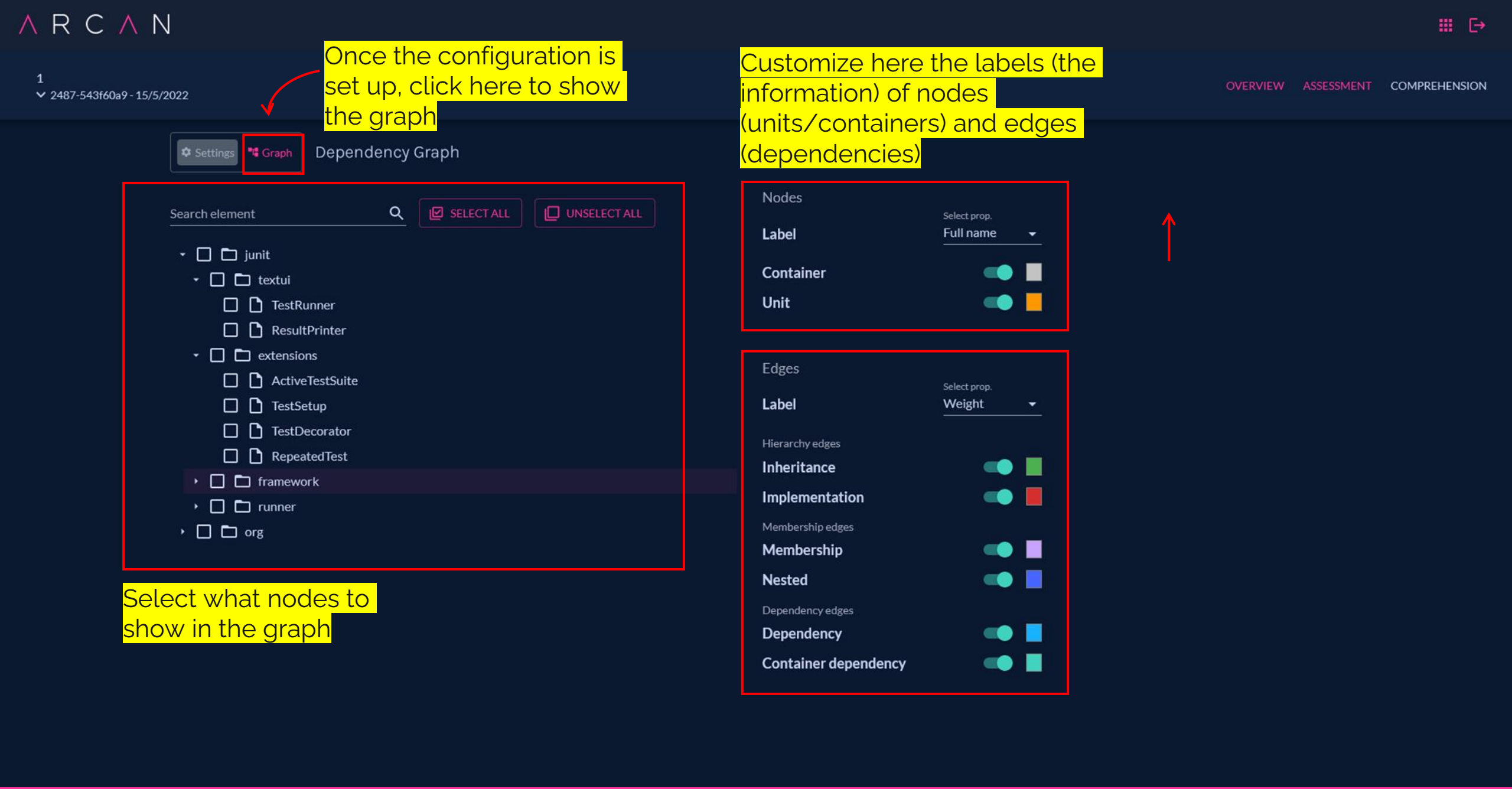
29.00
Threshold

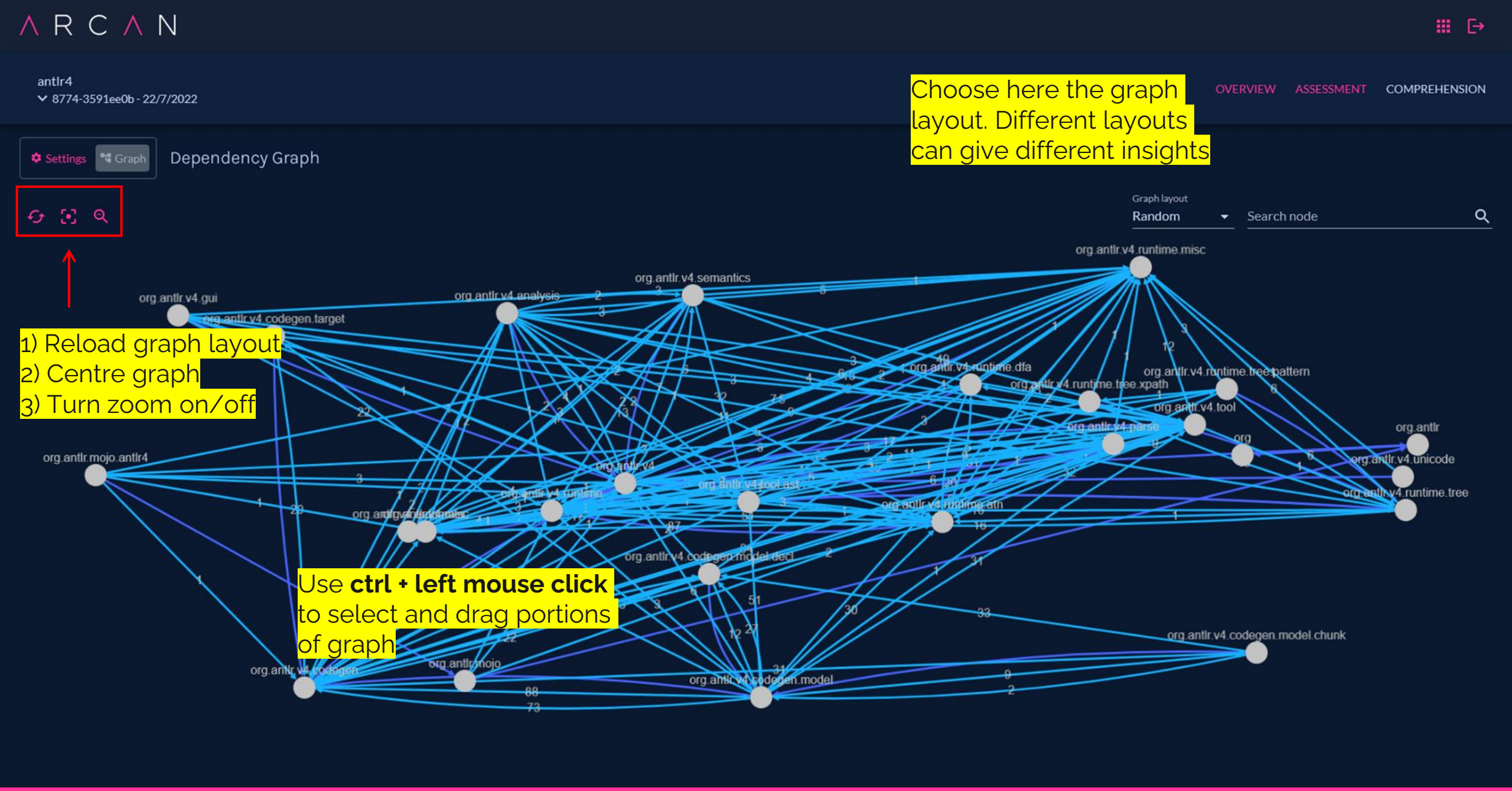
Fan In + Fan Out > System's median (29)
Detection rule


ARCAN
1
2487-543f60a9 - 15/5/2022
OVERVIEW
ASSESSMENT
COMPREHENSION
13902
Id
Hublike Dependency
Smell Type
300
TechDebt Index
8
Severity
Package
Affected Type
org.junit.runner
Affected Element(s)
21
Size
Graph layout
Circle
Search node
1) Reload graph layout
2) Centre graph
3) Turn zoom on/off
Change here the layout of the graph
Insert here the name of a unit or container to quickly find it in the graph
junit
org.junit.runners.model
org.junit.rule
org.junit.internal.builders
org.junit.runners.parameterized
junit.framework
org.junit.experimental.results
junit.runner
org.junit.internal.runners.model
org.junit.runner
org.junit.internal
org.junit.experimental.categories
org.junit.validator
org.junit.runner.notification
org.junit.internal.requests
org.junit.internal.runners
org.junit
Click either an edge or a node to show the source code
File
Dependant
Depends upon
Weight
Click either an edge or a node to view where dependencies are created.

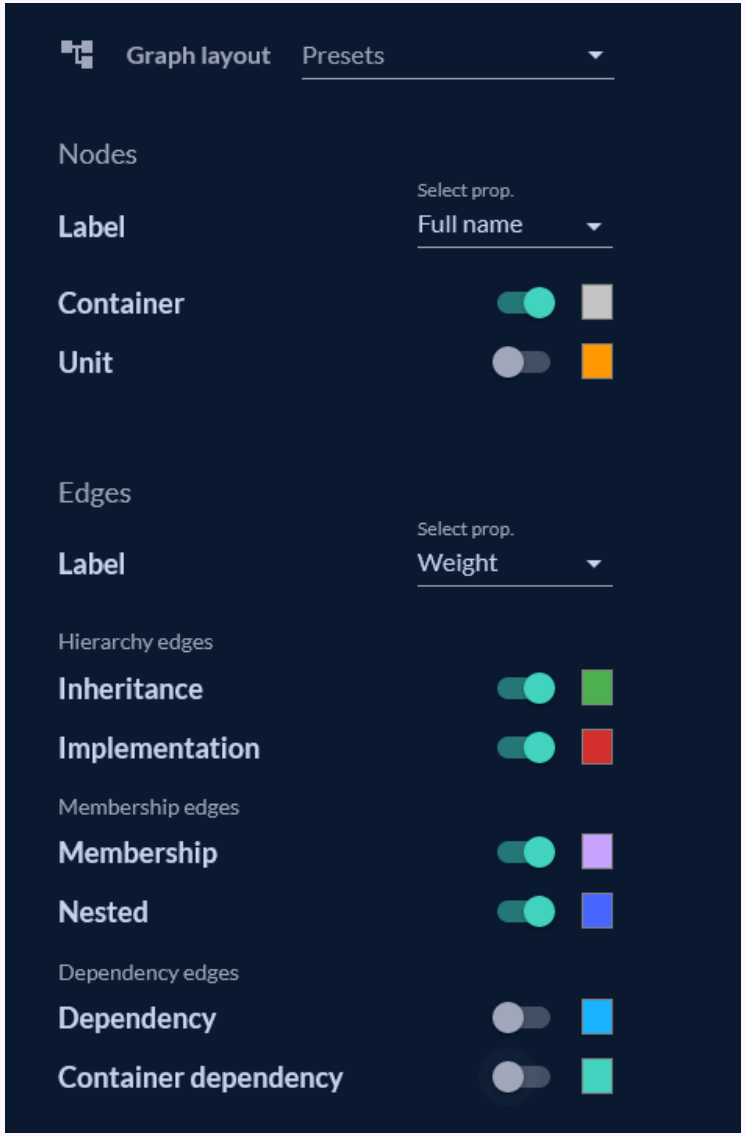
PAGE: **Dependency graph**

INSPECT A PROJECT

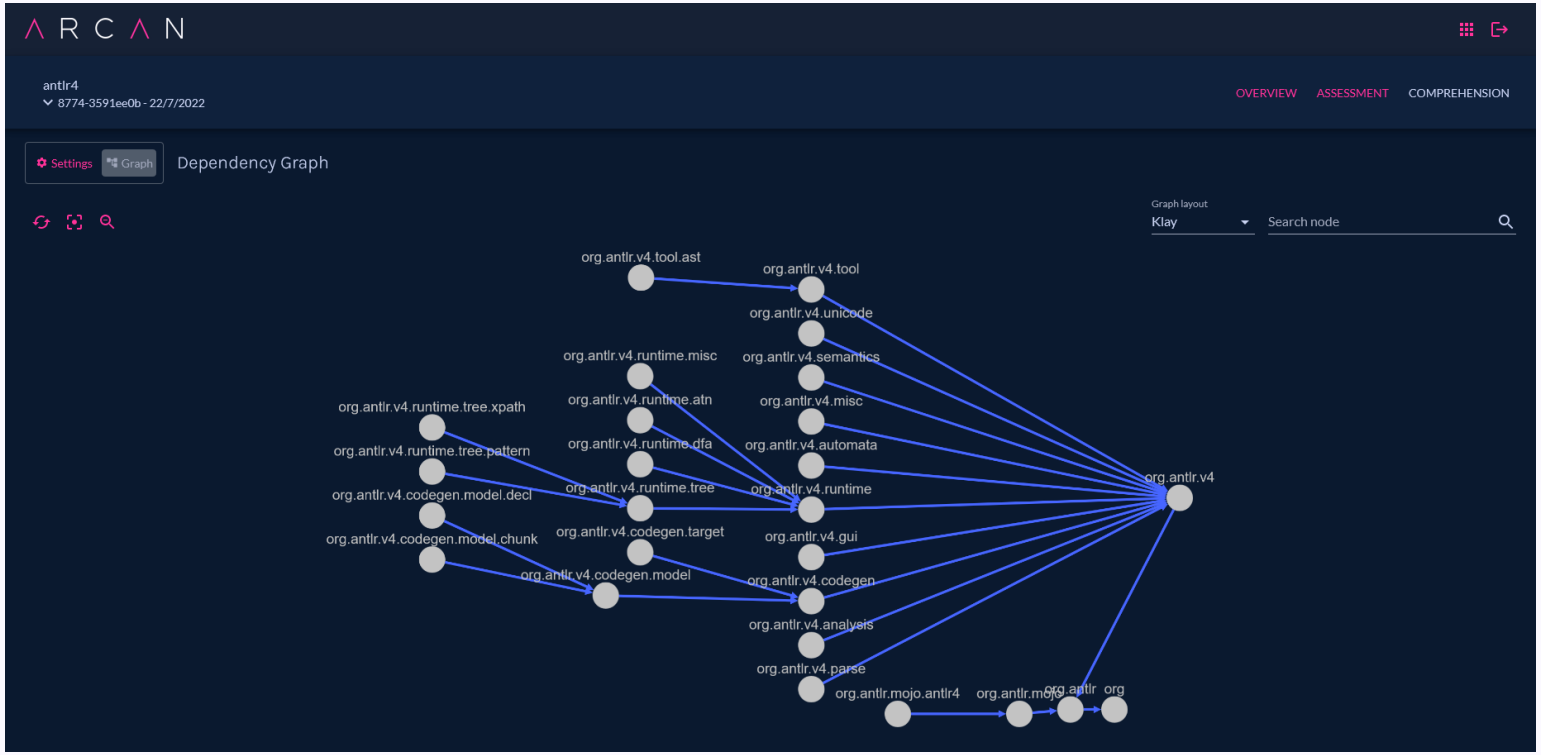

ARCAN
1
2487-543f60a9 - 15/5/2022
OVERVIEW
ASSESSMENT
COMPREHENSION
Once the configuration is set up, click here to show the graph
Settings
Graph
Dependency Graph
Search element
SELECT ALL
UNSELECT ALL
junit
textui
TestRunner
ResultPrinter
extensions
ActiveTestSuite
TestSetup
TestDecorator
RepeatedTest
framework
runner
org
Select what nodes to show in the graph
Customize here the labels (the information) of nodes (units/containers) and edges (dependencies)
Nodes
Label
Select prop.
Full name
Container
Unit
Edges
Label
Select prop.
Weight
Hierarchy edges
Inheritance
Implementation
Membership edges
Membership
Nested
Dependency edges
Dependency
Container dependency


ARCAN
antlr4
8774-3591ee0b - 22/7/2022
OVERVIEW
ASSESSMENT
COMPREHENSION
Settings
Graph
Dependency Graph
Graph layout
Random
Search node
Choose here the graph layout. Different layouts can give different insights
1) Reload graph layout
2) Centre graph
3) Turn zoom on/off
Use ctrl + left mouse click to select and drag portions of graph
org.antlr.v4.runtime.misc
org.antlr.v4.semantics
org.antlr.v4.gui
org.antlr.v4.codegen.target
org.antlr.v4.analysis
org.antlr.v4.runtime.dfa
org.antlr.v4.runtime.tree.xpath
org.antlr.v4.runtime.tree.pattern
org.antlr.v4.tool
org.antlr.v4.parse
org.antlr
org
org.antlr.v4.unicode
org.antlr.v4.runtime.tree
org.antlr.mojo.antlr4
org.antlr.v4
org.antlr.v4.runtime
org.antlr.v4.tool.ast
org.antlr.v4.runtime.atn
org.antlr.v4.codegen.model.decl
org.antlr.v4.codegen.model.chunk
org.antlr.v4.codegen
org.antlr.mojo
org.antlr.v4.codegen.model

Configuration

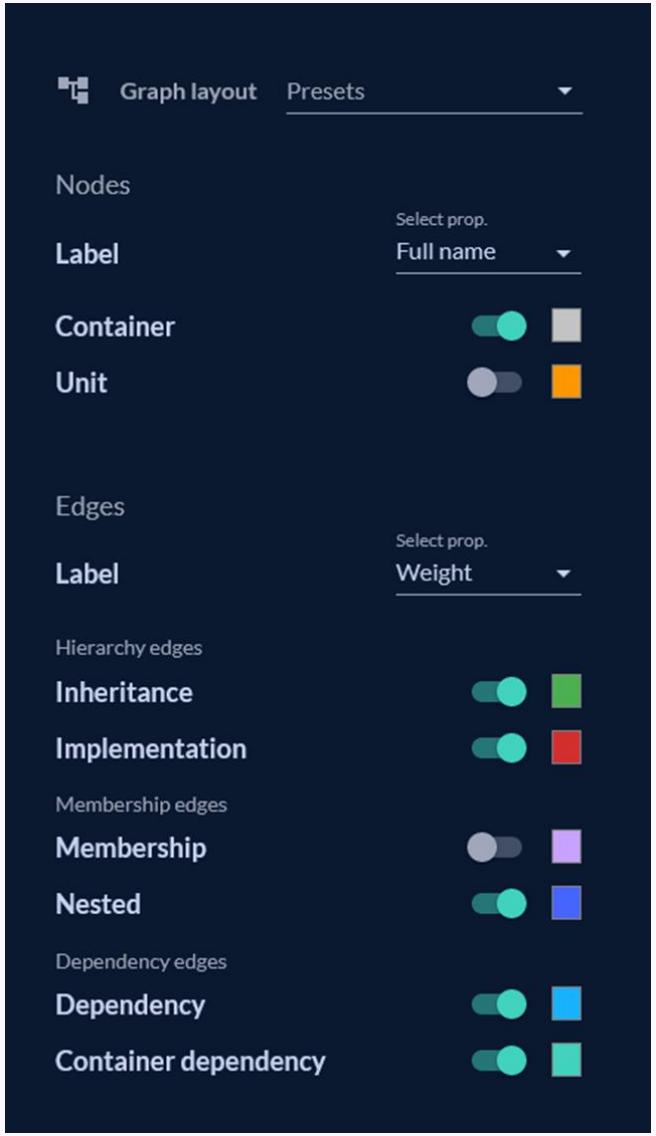


Graph layout: **KLAY**

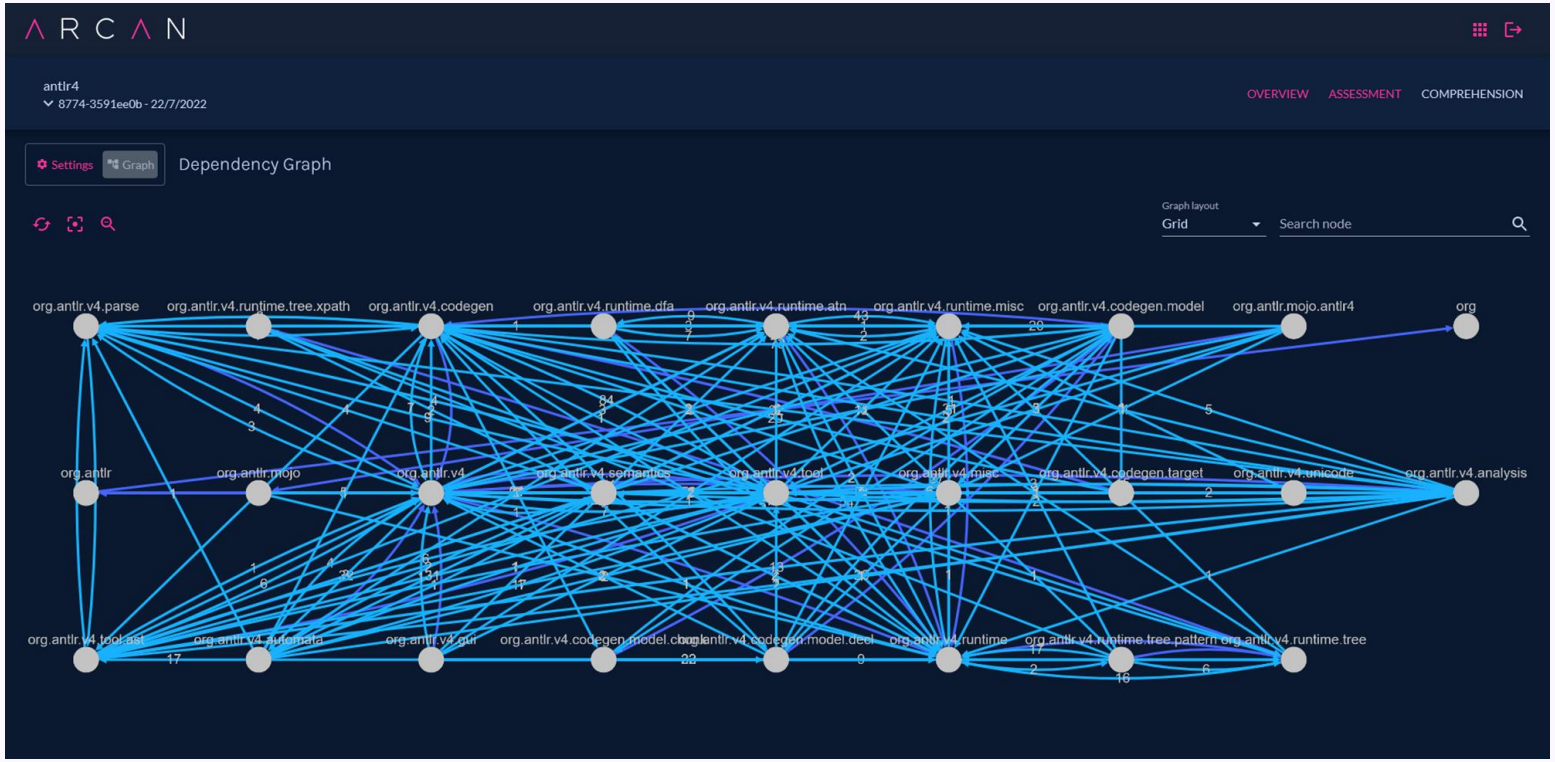


Useful to investigate how project's containers are nested one inside the others.

Configuration

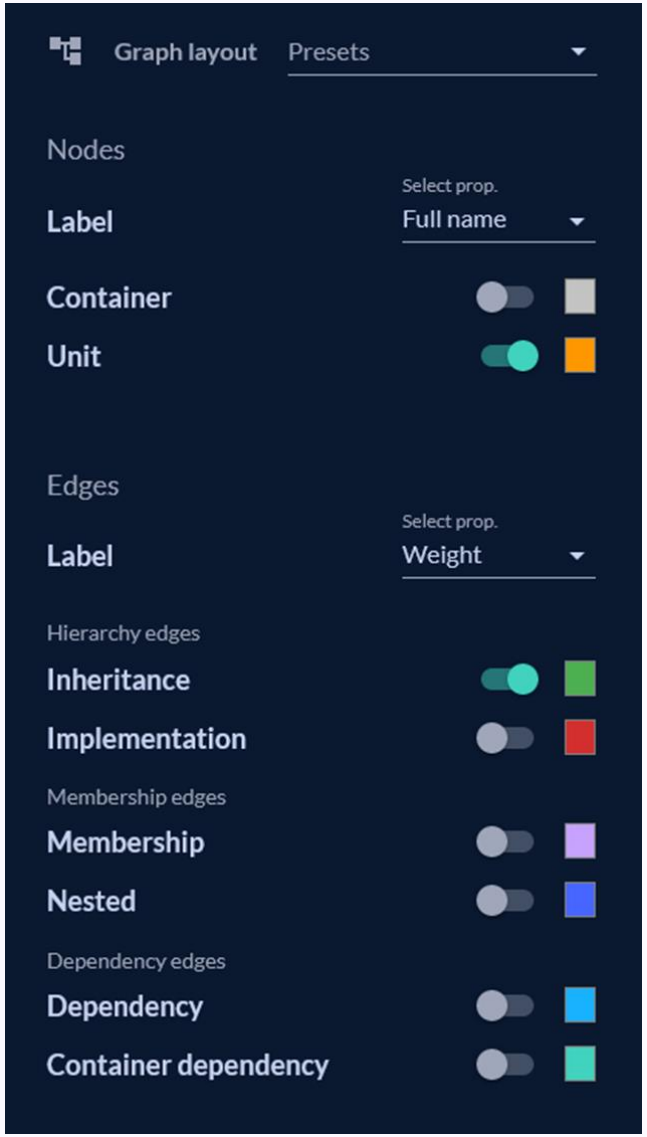


Graph layout: **GRID**

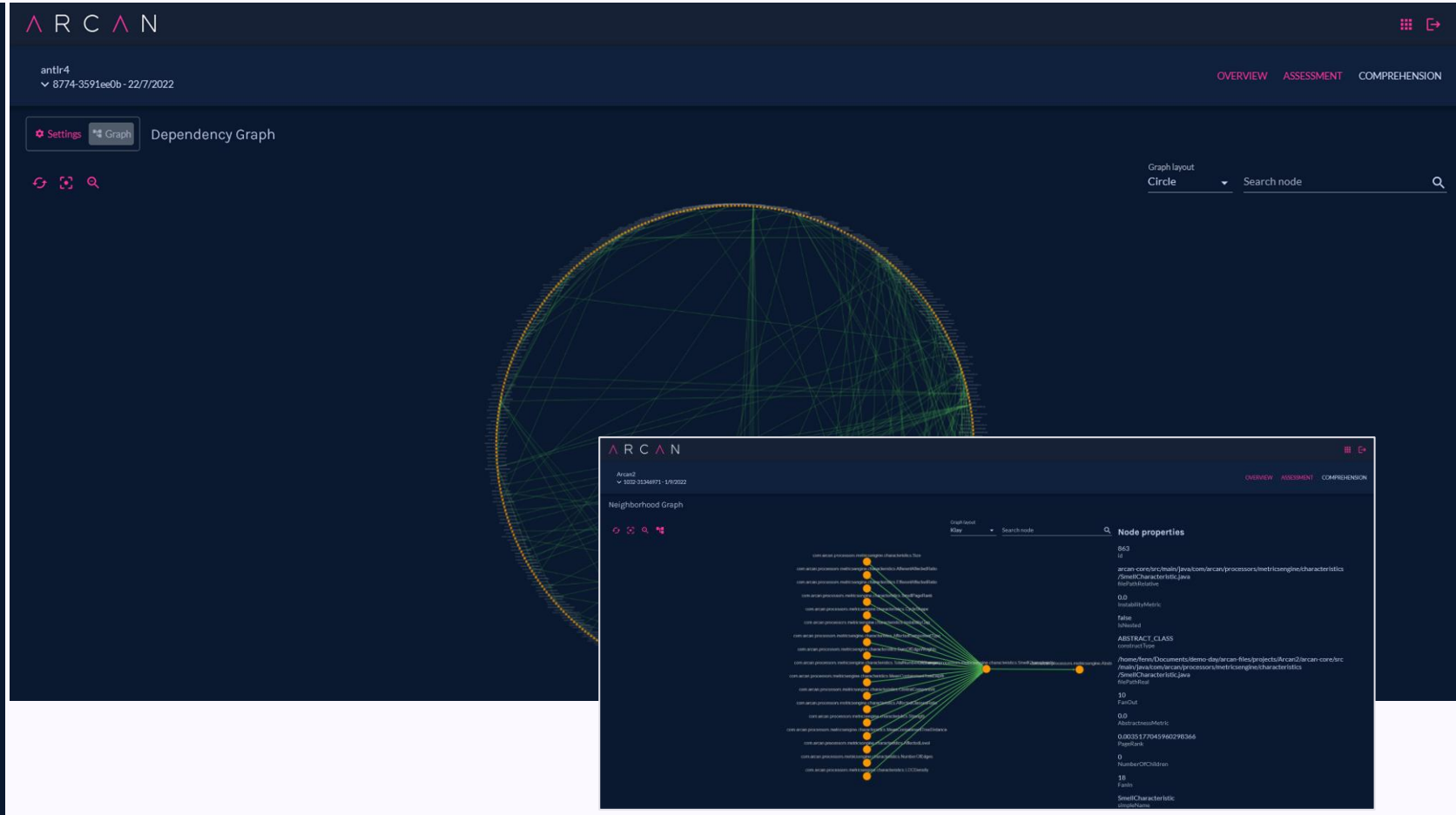


Useful to investigate the dependencies between different project's containers.

Configuration

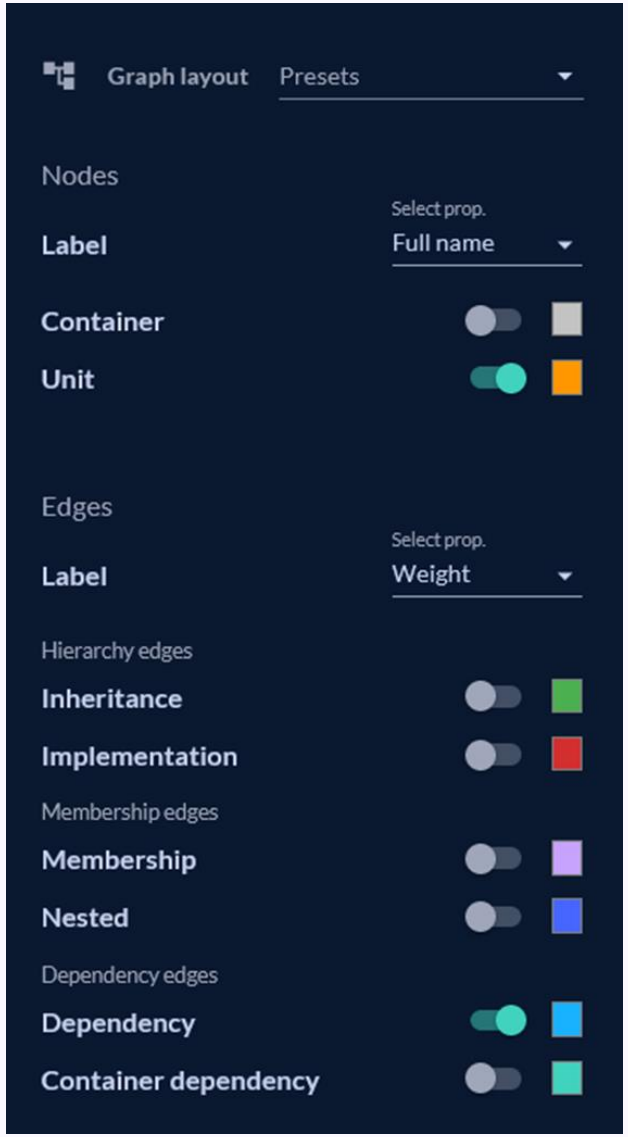


Graph layout: **CIRCLE**

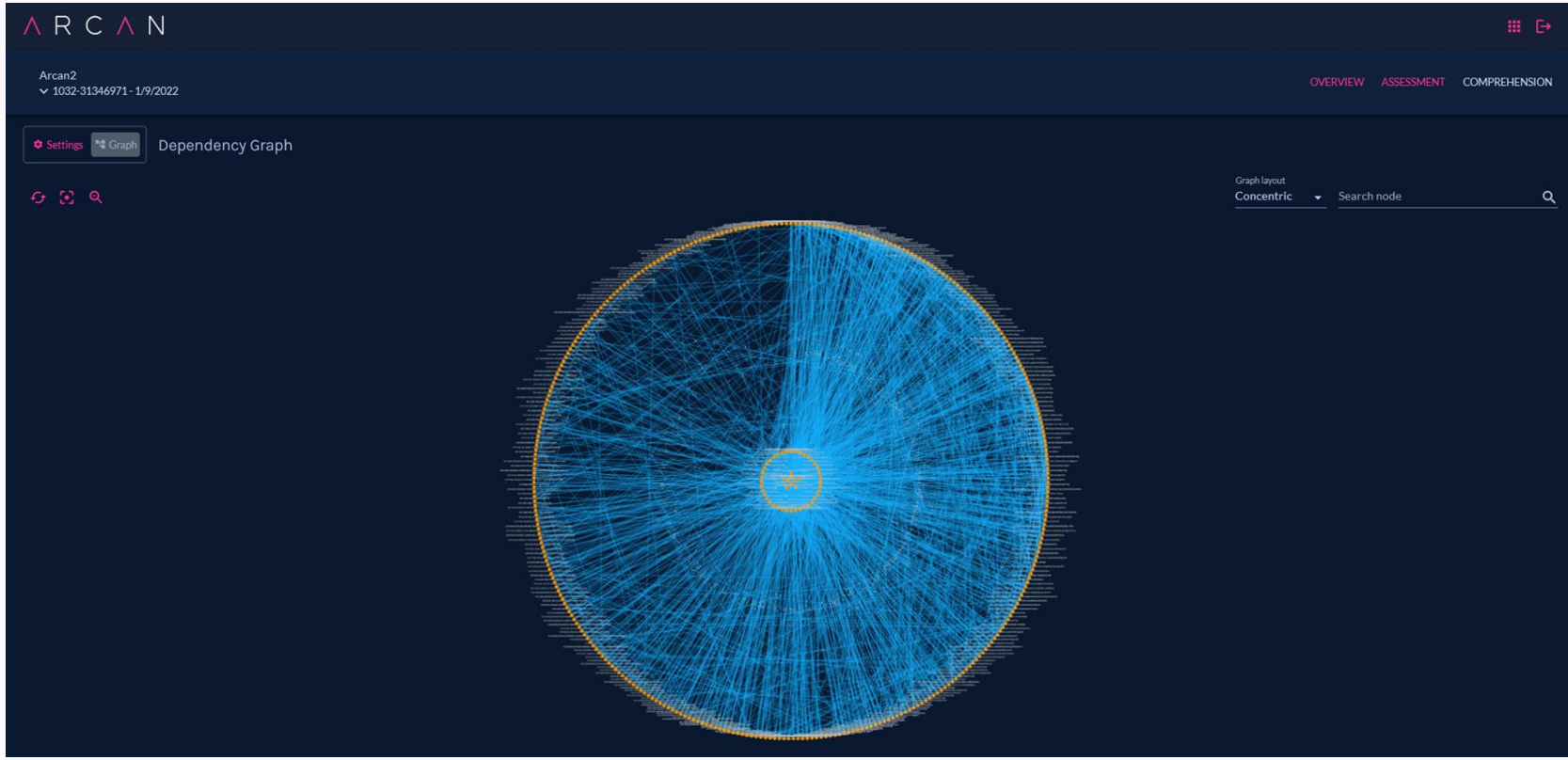


Useful to identify the abstractions of the main hierarchy structures. Click on the nodes that accumulates most of the dependencies to inspect the hierarchy.

Configuration



Graph layout: **CONCENTRIC**



Useful to identify the most used units of the project.

ARCAN

Do you want to report a bug or suggest a feature?
Create a new issue on Github:
https://github.com/Arcan-Tech/arcan-issues-public

info@arcan.tech
www.arcan.tech