PIACERE

Deliverable D4.6

IaC Code Security and components security Inspection - v3

Editor(s):	Matija Cankar (XLAB)
Responsible Partner:	XLAB d.o.o.
Status-Version:	Final v1.0
Date:	31.05.2023
Distribution level (CO, PU):	PU

Project Number:	101000162
Project Title:	PIACERE

Title of Deliverable:	IaC Code Security and components security Inspection - v3
Due Date of Delivery to the EC	31.05.2023

Workpackage responsible for the Deliverable:	WP4 - Verify the trustworthiness of Infrastructure as a code
Editor(s):	Matija Cankar (XLAB)
Contributor(s):	Grega Redek (XLAB), Anže Luzar (XLAB), Matija Cankar (XLAB)
Reviewer(s):	Juncal Alonso (Tecnalia)
Approved by:	All Partners
Recommended/mandatory readers:	WP2, WP3, WP5, WP8

Abstract:	This deliverable will present the outcome of Task T4.2 and Task T4.3. The deliverable comprises both a software prototype [KR6-KR7] and a Technical Specification Report. The document will include the Security Inspector technical design and implementation aspects. The document will also include the Security Inspector technical design and implementation aspects
Keyword List:	IaC, SAST, IaC Security, DevOps, DevSecOps
Licensing information:	This work is licensed under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
v0.1	31.03.2023	The ToC organisation	XLAB
v0.2	30.04.2023	First draft version	XLAB
v0.3	16.05.2023	Improved sections, ready for internal review	XLAB
v0.4	18.05.2023	Internal peer review	TECNALIA
v0.6	29.05.2023	Fixed all review requests	XLAB
v1.0	30.05.2023	Final quality check. Ready for submission	TECNALIA

Table of contents

List of tables

List of figures

Terms and abbreviations

CSP	Cloud Service Provider
DevOps	Development and Operation
DoA	Description of Action
EC	European Commission
GA	Grant Agreement to the project
IaC	Infrastructure as Code
IEP	IaC execution platform
IOP	IaC Optimization
KPI	Key Performance Indicator
PR or MR	Pull request or merge request
QA	Quality Assurance
SW	Software

Executive Summary

This is a third technical deliverable from a series of three, describing the progress and plans of the IaC Security Inspector and Component security inspector (KR6 and KR7). The mentioned results are output of tasks T4.2 and T4.3 from the WP4. The document is the last from series and provides a bit different table of content. First part is more oriented towards the presentation of the key results and the details including the integration with PIACERE DevSecOps framework are moved to appendix, which remains in the same structure as first two deliverables.

This document (as their predecessors) presents the advancements made in the IaC Security Inspector and Component Security Inspector (KR6 and KR7) over the course of the third year of the PIACERE project. The components are referred to as a unified entity, referred to as the IaC Scan Runner. The report provides an overview of the results from T4.2 and T4.3 in the main part, while the Appendix of this document includes user manual and developers notes. The main part consists of mainly new content in comparison to previous deliverables, while the Appendix is substantially revised and updated version of previous deliverables.

1 Introduction

This is the final technical deliverable from a series of three, describing the progress and plans of the IaC Security Inspector and Component security inspector (KR6 and KR7). The mentioned results are output of tasks T4.2 and T4.3 from the WP4. The deliverable integrates and updates content from the predecessors, therefore some content is duplicated. The differences are described in Section 1.2 Document structure.

The task T4.2 focuses on statical code analysis (SAST) of the IaC which will be generated from the PIACERE DOML (DevOps Modelling Language). The focus of this task is to find and create a set of useful code checks for different IaC options (e.g., Terraform HCL, TOSCA Simple Profile in YAML, Ansible…) and issues such as typos, syntactical problems, and secret management. An example for this is a check if the IaC include hard coded passwords.

The task T4.3 focuses on detecting the components used by the IaC. This means that IaC is inspected for the dependencies and their versions, which might have vulnerabilities like zero-day exploits. An example of this check would be if the OpenSSL library referred in IaC code is mature enough to not have a heart-bleed[1] vulnerability.

The approach of both tasks is different, but if we consider them as a black box, both have a similar behaviour – accepting IaC as an input and providing the list of vulnerabilities in the form of configuration errors, warnings, and suggestions in the output. Based on this observation we will develop and describe them jointly to ease the integration instructions and provide detailed instructions of how the checks can define the functionality of one or another.

1.1 About this deliverable

The deliverable presents the relevant aspects of the IaC Scan Runner component, which serves as a service to cover defined KR6 and KR7 in PIACERE project. The deliverable explains the overview of the final key results, the early experiments and lessons learned during the fulfilment of our goal of finding the vulnerabilities in the deployable applications and therefore providing the better IaC without any security issues. The document presents the final architecture plan and requirements fulfilment status, according to the requirement list defined in Deliverable D2.2 [1]

1.2 Document structure

The introduction Sections were updated and present the document content and structure. The following sections from 2 to 5 are new, while the appendix is comprised of the updated version of the previous two deliverables. In the rest of the document we cover:

- Section 2: presents the *KR6 and KR7 overview*, namely the IaC Scan runner, functional requirements, recent changes and updates, where we can find the most recent online information.
- Section 3: Focuses on preliminary results.
- Section 4 points out the lessons learned and
- Section 5 concludes the document.

2 KR 6 and KR 7-ISR overview

The KR 6 and KR 7 are two different IaC inspection tools, that are combined and sometimes overlapped in the result called IaC Scan Runner (ISR). According to the configuration, IaC Scan Runner can serve as IaC security inspector (KR 6) or component security inspector (KR7) or both at the same time.

The IaC scan runner is used as a SAST tool after the IaC design step, when we would like to inspect the designed IaC template for security issues. The tools give to the DevSecOps developer an opportunity to initiate multiple security SAST checks simultaneously.

More details about the IaC Scan Runner can be found in the appendix of this document (covering the architecture and integration to the PIACERE DevSecOps framework), while most up-to-date content is available on the:

- Documentation page: https://xlab-si.github.io/iac-scanner-docs/
- Documentation repository: https://github.com/xlab-si/iac-scanner-docs
- Project repository: https://github.com/xlab-si/iac-scan-runner

Beside the information provided in the open-source repositories, the tool has been mentioned and described in scientific articles [2] [3] [5].

2.1 Changes in v3

In the last development cycle, we focused on improving the tools according to the use-case requests and feedback from early adopters and testers. The changes are mainly focused on improved usability and the clearness of the tool usage. The changes in the last year are threefold, namely:

Multi-project approach: we have implemented a new feature that allows users to configure settings for each of their projects individually. This enables them to set specific parameters and preferences for each project, giving them complete control over their workflow and allowing them to customize the system to suit their needs.

Performance and security improvements: we have improved the performance and security of the system by updating the API endpoints, which makes it faster and more efficient for users to access and use the API.

Improved user experience: we have given the system's user interface a more modern design, which includes new design elements and features that make it more visually appealing and user-friendly. This makes it easier for users to navigate through the system with ease.

The changes resulted in 13 new pull requests (PRs), 2 new versions of the software component and a reconfigured API for a more convenient IDE user interface where the IaC Security Inspector and Component security inspector are integrated. Details can be found in the GitHub[1] page or in the Appendix of this document.

2.2 Functional description and requirements coverage

Through the PIACERE consortium collaboration, we defined a set of necessary requirements for static code inspection, that are available in the PIACERE architectural specification [1] and includes integrational requirements and use-case requirements. These requirements drove our development and research through the whole project.

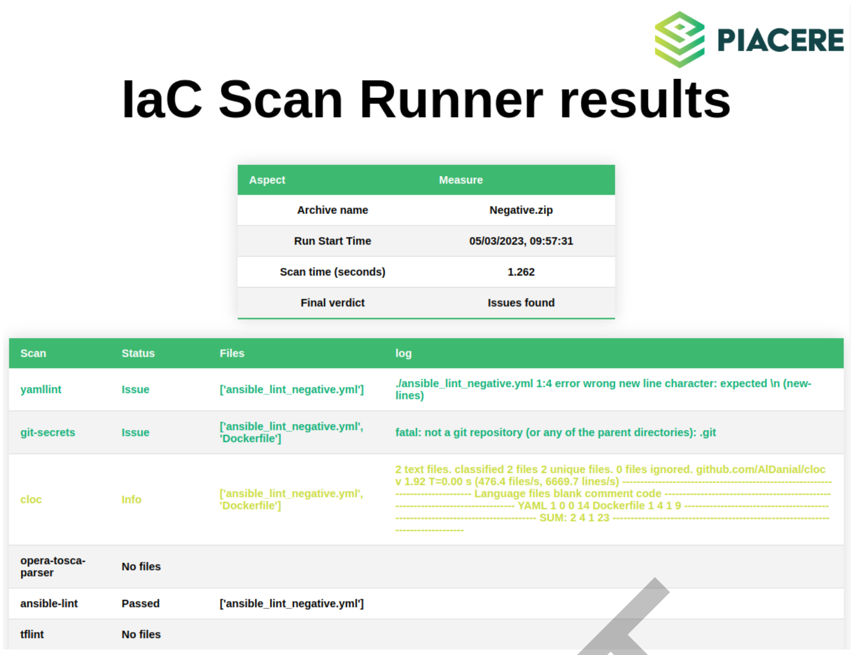
[1] https://github.com/xlab-si/iac-scan-runner

Table 1: Requirements for KR6 and KR7

REQ ID	Description	Complexity /Task	Acceptance	Priority	Status
REQ23	IaC Code Security Inspector must analyse IaC code w.r.t. security issues of the modules used in the IaC.	Medium T4.2	**ACCEPTED**	MUST HAVE	**Done**
REQ24	Security Components Inspector must analyse and rank components and their dependencies used in the IaC.	Medium T4.3	**ACCEPTED**	MUST HAVE	**Done**
REQ65	IaC Security Inspector and Component Security Inspector should hide specificities and technicalities of the current solutions in an integrated IDE.	Low T4.2, T4.3, WP3f	**ACCEPTED**	MUST HAVE	**Done**
REQ66	IaC Code security inspector must provide an interface (CLI or REST API) to integrate with other tools or CI/CD workflows.	Medium T4.2, T2.2	**ACCEPTED**	MUST HAVE	**Done**
REQ67	IaC Component security inspector must provide an interface (CLI or REST API) to integrate with other tools or CI/CD workflows.	Medium T4.2, T2.2	**ACCEPTED**	MUST HAVE	**Done**
REQ80	SAST tools to check Docker configurations shall be included in the Canary environment.	Medium WP4	**ACCEPTED (Re-worded)**	MUST HAVE	**Done**

As can be seen in Table 1 all requirements were satisfied inside the Y3 version of the tool. Each individual requirement has been fulfilled by the following actions:

- **REQ23**: IaC Scan Runner takes care of analysing IaC code w.r.t security issues of the modules used in the IaC with help of the following integrated checks (see Table 2).
- **REQ24:** The fulfilment of this requirement is twofold. First, we included a set of scans that investigate the components and dependencies used in the IaC code (see KR7 column in Table 2). Secondly, for **ranking**, we have upgraded our system by implementing a code scanner that checks multiple files and categorizes the scan results based on their status. The status categories include **Issue** (requires user intervention), **Info** (includes notifications for the user), **Passed** (nothing to do), and **no files** (for the specific check no files were find). This allows you to quickly determine the status of each scanned file and prioritize your actions accordingly. Additionally, we have introduced a new output (JSON and HTML – see Figure 1) that presents the scan results grouped by status in a **descending order of severity**. This page provides an **overview of the scan outcomes** and helps in prioritizing efforts based on the gravity of the identified issues. The page is divided into four sections that correspond to the status of the files: ISSUE, INFO, PASSED, and NO FILES. For each status category, the page displays a list of relevant files with a brief summary of the issue (if any), which makes it easy to identify critical files and take prompt action.



PIACERE

IaC Scan Runner results

Aspect	Measure
Archive name	Negative.zip
Run Start Time	05/03/2023, 09:57:31
Scan time (seconds)	1.262
Final verdict	Issues found

Scan	Status	Files	log
yamllint	Issue	['ansible_lint_negative.yml']	./ansible_lint_negative.yml 1:4 error wrong new line character: expected \n (new-lines)
git-secrets	Issue	['ansible_lint_negative.yml', 'Dockerfile']	fatal: not a git repository (or any of the parent directories): .git
cloc	Info	['ansible_lint_negative.yml', 'Dockerfile']	2 text files. classified 2 files 2 unique files. 0 files ignored. github.com/AlDanial/cloc v 1.92 T=0.00 s (476.4 files/s, 6669.7 lines/s) --- Language files blank comment code --- YAML 1 0 0 14 Dockerfile 1 4 1 9 --- SUM: 2 4 1 23 ---
opera-tosca-parser	No files		
ansible-lint	Passed	['ansible_lint_negative.yml']	
tflint	No files		

Figure 1: A screenshot of IaC Scan results formatted in HTML

- **REQ65:** This requirement is achieved through the IaC Security Inspector and Component Security Inspector modules of the API. These modules **work seamlessly with integrated development environments** (IDEs), allowing developers to leverage the benefits of a user-friendly and **intuitive interface** without being bogged down by the technical complexities of the security solution. This not only increases efficiency but also ensures a consistent and streamlined approach to security across the development lifecycle.
- **REQ66, REQ67:** Both requirements are addressed through the **API's interface options**. The IaC Code Security Inspector module provides an interface in the form of a Command Line Interface (CLI) or REST API that developers **can easily integrate with other tools** or CI/CD workflows. This flexibility allows for the seamless integration of security checks into the development pipeline, ensuring that vulnerabilities and issues are identified and resolved early in the process. The IaC Component Security Inspector module also provides an interface in the form of a **CLI or REST API**. This interface allows organizations to easily integrate component security checks into their existing tools and workflows, further enhancing their security posture.
- **REQ80:** During the project it has been realised that SAST inspections can be already done in IaC Scan Runner. To fulfil this requirement we integrated Hadolint IaC Check into the list of scans, which means that IaC is already checked before the Canary Environment is created.

Table 2: KRs coverage by IaC Scan Runner checks.

IaC Check	Target IaC entity	Security (KR6)	Component (KR7)	Other
xOpera TOSCA parser	**TOSCA**			✓
Ansible Lint	**Ansible**			✓
Steampunk Spotter	**Ansible**	✓	✓	
TFLint	**Terraform**			✓

tfsec	**Terraform**	✓		
Terrascan	**Terraform**	✓		
yamllint	**YAML**			✓
Pylint	**Python**			✓
Bandit	**Python**			
Safety	**Python packages**	✓	✓	
Gitleaks	**Git repositories**	✓		
git-secrets	**Git repositories**	✓		
Markdown lint	**Markdown files**			✓
hadolint	**Docker**			✓
Gixy	**Nginx configuration**	✓		
ShellCheck	**Shell scripts**	✓		✓
ESLint	**JavaScript**			✓
TypeScript ESLint	**TypeScript**			✓
HTMLHint	**HTML**			✓
stylelint	**CSS and other styles**			✓
Checkstyle	**Java**			✓
cloc	**Multiple components**			✓
Snyk	**Multiple components**	✓	✓	✓
SonarScanner	**Multiple components**	✓	✓	✓

2.3 Main innovations

In today's fast-paced software development landscape, Infrastructure as Code (IaC) has emerged as a critical technology for creating, deploying, and managing cloud infrastructure. IaC enables developers to define infrastructure components such as servers, networks, and storage in code, making it easier to provision, configure, and manage infrastructure at scale. However, like any code, IaC can have issues, vulnerabilities, and best practice violations that can compromise the security, compliance, and efficiency of the infrastructure.

The main innovation opportunities of the IaC development is controlling the code itself and keeping it safe and secure. DevOps approach gives as the agility of evolving and fixing the infrastructure code, however, when the project is large, complexity raises and teams need to rely on tools. The main challenge is correctness of the code, meaning, we deploy and set only the things that we want and need. The second one is continuous (daily) inspection, which means that code inspection actions are not limited only to the application design time, but to the runtime as well, because some used components will become vulnerable in the future. Another important challenge will become a reality with new AI and *ChatGPT* era. The idea of generating code with AI tools is very attractive, but also very dangerous as it can leave in the code unexplained facts.

To address these challenges we empower developers and DevOps teams with a tool that can scan IaC code and identify potential issues before they become major problems. That's where the IaC Scan Runner comes in.

One step for multiple tools: IaC Scan Runner is a powerful tool that enables developers and DevOps teams to scan IaC code with multiple tools and identify issues, vulnerabilities, and best practices violations. IaC Scan Runner supports various scanning tools such as Hadolint, Steampunk Spotter, Sonar, and more, providing a comprehensive view of the IaC code's quality and security. With IaC Scan Runner, you can quickly identify issues, prioritize them based on severity, and take necessary actions to fix them, ensuring that your infrastructure is secure, compliant, and efficient.

Customise your scan per project and generate report: IaC scan runner works by scanning IaC code with multiple tools and beside individual scan report, it generates a comprehensive combined report of the scanning results. The scanning process is highly customizable, with per-project configuration options that allow you to customize the scanning parameters to suit your specific needs and requirements. For example, you can choose the tools used for scanning, the ruleset, the severity levels, and more, ensuring that IaC Scan Runner scans your IaC code based on your organization's policies and standards.

Easy to integrate with your setup: Once the scanning process is complete, IaC Scan Runner generates reports in either **JSON** or **HTML** format, depending on your preference. The JSON report provides a detailed view of the scanning results, including the issues found, the severity level, and the location of the issue in the code. The HTML report has a modern design that is easy to read and navigate, with clear visualizations and actionable insights. You can quickly identify the issues that require attention, prioritize them based on severity, and take necessary actions to fix them. Also note that beside **REST** approach, IaC Scan Runner comes also with a **command-line interface (CLI)** that makes it easy to integrate with your existing workflows and toolchains defined in scripts. You can use the CLI to automate the scanning process, schedule scans, and incorporate IaC Scan Runner into your DevOps pipeline, ensuring that your IaC code is continuously scanned for issues and vulnerabilities.

Open-source tool with premium features: The IaC Scan Runner an open-source tool that has integrations with other open-source tools and also proprietary and paid services. This allows to be used as a tool with basic and premium features, that provides an easier adoption to the existing environments that grow with the size and maturity of the target project.

3 Overview of preliminary experiments

The IaC Scan Runner has been tested and demonstrated to the PIACERE use cases, a commercial DevOps tools developers of XLAB Steampunk[2] team, some other security projects (as ICOS H2020 [6]) and PIACERE community over the YouTube.

From the use-cases we have received quite a few recommendations during the development, mostly focusing on the application coverage (new scans) and readability of results. The UX of results have been lately substantially improved, and use-cases had expressed the clearness of the report and appropriate ranking.

The checks are sufficient and for each use case we improved some particularities, e.g. for Public Administration use-case the Ansible checks were tailored, for Maritime solutions the Terraform checks and for Ericsson the coverage of docker security checks was in focus.

An important result for DevSecOps practitioners is to encapsulate more checks into a single service, saving time for using the individual tools. The benefits of the IaC Scan Runner have been observed also by other projects, e.g. ICOS, that might include the IaC Scan Runner service in their security SAST workflow.

Overall, the IaC Scan Runner has proven to be a comprehensive approach to scanning Infrastructure as Code (IaC). The tool's ability to scan IaC using multiple tools has been particularly appreciated, as ensures that the results are accurate, and service has multiple options for integration (REST or CLI). Users also appreciate IaC Scan Runners user-friendly interface and the range of output formats available for presenting the scan results, including HTML and JSON. The API that supports per project configuration has been particularly helpful for teams that work on multiple projects with different requirements.

[2] https://steampunk.si/

4 Lessons learnt and outlook to the future

The development of IaC Scan Runner has been a fruitful endeavour with a constant collaboration with users and improvement of things for the better. We learned that it was very successful to start first with the inquiry of users, what they do need and to make a review of the market with the available open-source and on the shelf solutions.

Combining the power of the existing solutions have provided us the crucial insight. What can be already covered and what we are still missing. This part led us to change the component scanner in a way to focus more on IaC components and dependencies, where we found most of the interest of users. This initial solution and user base provided us an understanding of the field and a market gap on the component check tools for Ansible.

We see that IaC Scan Runner has gained some interest in other projects and can be used as a companion to all other scanning tools. As the field of deployment and automation is expanding from cloud to computing continuum integrating more and more heterogeneous components and services, the need for scanning the correctness and safety will become even more desired.

The future research of the IaC security is still very challenging, interesting, and agile. Automation through IaC is popular, but it has not gained enough trust that enterprises would jump into changing all their solutions, especially critical ones, to follow the full DevSecOps approach. To fulfil the market needs in the future, we will continue with the integration of new checks in the IaC Scan runner and development of the integrated tools. The emphasis will be in component checks that are very desired nowadays – one specific component check is Steampunk Spotter, which just recently entered the market and attract new users that will provide market related requirements for future development.

5 Conclusions

The deliverable – v3 – has accomplished to conclude the story initiated and continued from the deliverable v1 and v2 with the same name. The first two deliverables presented the initial PoC development and the improvements in the year 1 and 2 respectively. The current one presents more general aspects of the tool in the sense of early experiments and main innovations, followed by the lessons learned and future work.

The appendix of this documents includes the technical details, developer notes and user manual of the components. The future work of this component can be followed on the corresponding Git repositories expressed in the introduction Section. The last details covering PIACERE and IaC Scan Runner, focusing on integration in the PIACERE IDE and PIACERE framework will be presented in the last technical deliverable "PIACERE DevSecOps framework v3" and updated also on the online documentation pages.

6 References

[1] E. Morganti, "D2.2 PIACERE DevSecOps Framework Requirements specification, architecture and integration strategy – v2," Zenodo, 2023.

[2] N. Petrović, M. Cankar and A. Luzar, "Automated Approach to IaC Code Inspection Using Python-Based DevSecOps Tool," in *2022 30th Telecommunications Forum (TELFOR)*, Belgrade, 2022.

[3] M. C. A. Luzar and G. Celozzi, "D4.4-IaC Code security and components security inspection-v1.0," 2021. [Online]. Available: https://www.piacere-project.eu/sites/d8piacere/files/Deliverables/D4.4-IaC%20Code%20security%20and%20components%20security%20inspection-v1_V1.0_20211130.pdf. [Accessed 21 November 2022].

[4] N. Petrovic, "ChatGPT-Based Design-Time DevSecOps," 2023.

[5] J. Alonso, P. Radoslaw and C. Matija, "Embracing IaC through the DevSecOps philosophy: Concepts, challenges, and a reference framework," *IEEE Software,* 2022.

[6] I. Consortium, "ICOS H2020 Web page," 2023. [Online]. Available: https://www.icos-project.eu/.

[7] "Swagger UI," [Online]. Available: https://swagger.io/tools/swagger-ui/. [Accessed 20 November 2022].

[8] "Python Package Index (PyPI)," [Online]. Available: https://pypi.org/project/iac-scan-runner/.

[9] "IaC Scan Runner Docker image," [Online]. Available: https://hub.docker.com/r/xscanner/runner.

[10] "Common Weaknes Enumeration," [Online]. Available: https://owasp.org/www-project-dependency-check/. [Accessed 10 2021].

[11] E. Morganti, A. Motta, L. Blasi, C. Nava and C. Bonferini, "D2.1 PIACERE DevSecOps Framework Requirements specification, architecture and integration strategy - v1," 2021.

[12] "PyMongo 4.3.3 Documentation," [Online]. Available: https://pymongo.readthedocs.io/en/stable/. [Accessed 21 November 2022].

[13] "MongoDB," [Online]. Available: https://www.mongodb.com/. [Accessed 21 November 2022].

APPENDIX: Implementation, delivery and usage

1 Implementation

1.1 Fitting into overall PIACERE Architecture

The *IaC Security Inspector* and *IaC Component Security Inspector* are a part of the *PIACERE Vulnerability tools*. The Vulnerability tools are used to check DOML, which is achieved with *Model checker* and check the IaC generated from the DOML, which is done by IaC Security Inspector and IaC Component Security Inspector.

In the overall architecture, the IaC Security Inspector and IaC Component Security Inspector fit into a set of design tools. The services are initiated by IDE after IaC will be generated from the DOML language. The **inputs** of the IaC Security Inspector and Component inspector are simplified to allow scanning IaC packages (such as zip or tar files). The **outputs** are be formatted as JSON and will be sorted by tools.

The main integration point is a RESTful API that will allow scanning IaC for issues and vulnerabilities. Another possible integration that can be established through the CLI, which offers the integration in console environments. This facilitates running the API within shell or interacting with it using different CLI commands. The API is encapsulated in a public Docker image, which makes it possible to run across all platforms. The different tools and services for IaC scanning have documentation of how each of these tools can be used and configured to fit the user's expectations. Future implementations may also introduce Software as a Service component that will allow users to organize their scans in a multi-workspace environment. Here we could provide the SaaS API, CLI, GUI and a possible standalone Eclipse plugin for a smoother integration.

1.2 Technical description

In this section we will present the IaC security inspector and Component security inspector in detail. The sequence diagram in Figure 2 presents activities of both tools. The standard IaC Security Inspector workflow starts with the user that desires to inspect his IaC with triggering the service directly from the PIACERE IDE. Within the IaC code inspection process, the IaC inspector initiates and runs the necessary checks (linters, configuration checks) using its internal worker. After that the inspector obtains the check results and returns them back to the user.

The Component Security Inspector has different task, which focuses on finding vulnerabilities in IaC dependencies (e.g., Python packages). As shown on the second block in Figure 2 the Component Security Inspector initiate component checks and seeks for component issues and misconfigurations. After getting the job done, the component creates an output and sends it back to IDE.

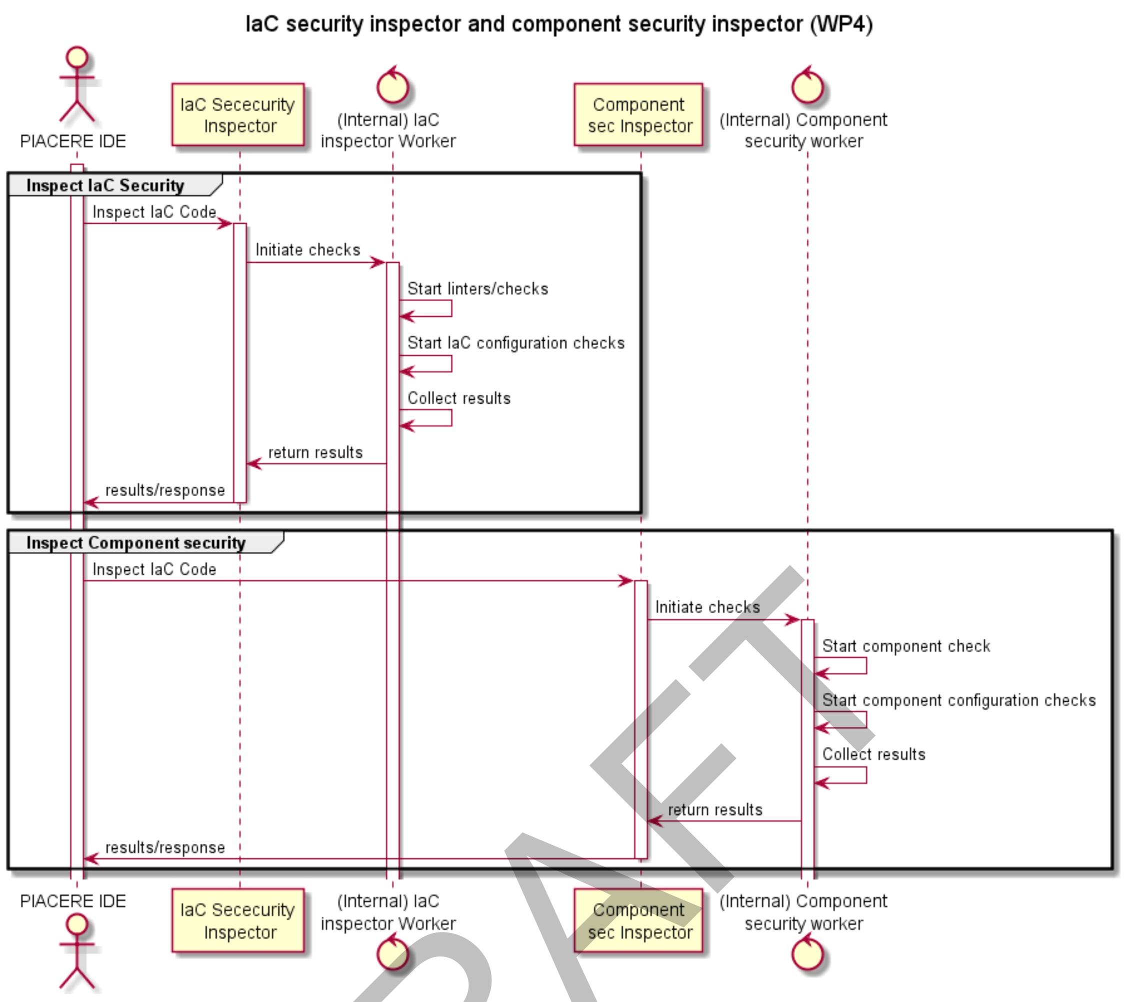


Figure 2: Simplified sequence diagram of IaC Security and IaC component inspectirs.

As it is evident from the sequence diagram in Figure 2, both tools have the similar interface and similar basic calls are required for managing the checks, starting checks, and stopping the checks. From the integrational perspective, the tools can be developed within one universal interface that can cover requirements of both tools. During the security check and security scan service review, we realised that some checks can also perform both types of checking - the IaC and component one. That means that a particular check could act as an IaC Security inspector or IaC component inspector or in case of more comprehensive checks – it can belong to both component types. This led us to the unification of the development of core element for both components, which we called as *IaC Scan Runner*.

IaC Scan Runner is an individual component that can run IaC Security Inspector or IaC Component Inspector checks included in scans. In other words, users interacting with the component can use it as any combination of both tools – only IaC scans, only component scans, and combined. The choice of how this component acts is defined by the list of enabled checks performed over the IaC.

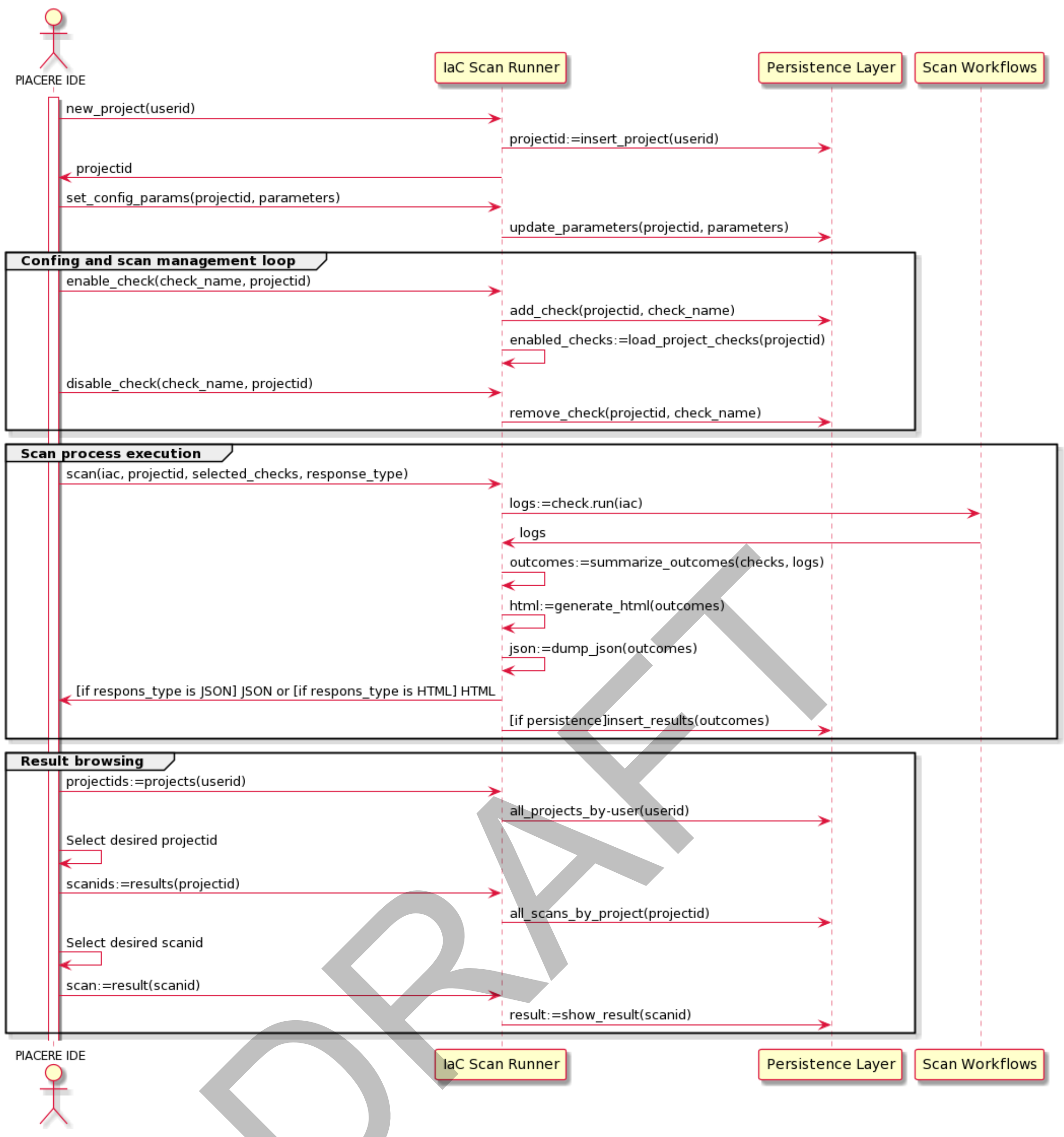

PIACERE IDE
IaC Scan Runner
Persistence Layer
Scan Workflows
new_project(userid)
projectid:=insert_project(userid)
projectid
set_config_params(projectid, parameters)
update_parameters(projectid, parameters)
Confing and scan management loop
enable_check(check_name, projectid)
add_check(projectid, check_name)
enabled_checks:=load_project_checks(projectid)
disable_check(check_name, projectid)
remove_check(projectid, check_name)
Scan process execution
scan(iac, projectid, selected_checks, response_type)
logs:=check.run(iac)
logs
outcomes:=summarize_outcomes(checks, logs)
html:=generate_html(outcomes)
json:=dump_json(outcomes)
[if respons_type is JSON] JSON or [if respons_type is HTML] HTML
[if persistence]insert_results(outcomes)
Result browsing
projectids:=projects(userid)
all_projects_by-user(userid)
Select desired projectid
scanids:=results(projectid)
all_scans_by_project(projectid)
Select desired scanid
scan:=result(scanid)
result:=show_result(scanid)
PIACERE IDE
IaC Scan Runner
Persistence Layer
Scan Workflows

Figure 3: IaC Scan Runner archive scan workflow with persistence and configurations

1.2.1 Prototype architecture

The previously introduced IaC Scan Runner is a *technical* component that can run *security scans*, while the *type of the scan* defines which the component type (IaC Security Inspector or IaC Component inspector). The IaC Scan Runner component diagram in Figure 4 presents the idea in the current prototype development, and Figure 3 presents the sequence diagram, showing the main user scenarios. The IaC Scan Runner is developed as a service inside the docker container. Basically, the service provides an API for configuring scans, managing scans, and retrieving outputs. The configuration manager keeps the configuration of each check, the Scan worker sets up the scan workflows according to the documentation and executes the scans. The processes inside a container are performing scans one by one. When the scans are finished, the service combines the output and sends it back to the IDE.

The idea of hiding complexity of the IaC Security Inspector and Component Security Inspector inside the IaC Scan Runner is done intentionally to ease the tool integration inside the PIACERE solution. Beside relying on the integrated checks, we envision also to run third-party remote scan services that are available to the user through free or paid remote services,e.g., like Snyk and Spotter which are already intergrated). These comprehensive services can include any possible checks from linters, QA, security or component check provided by a third party.

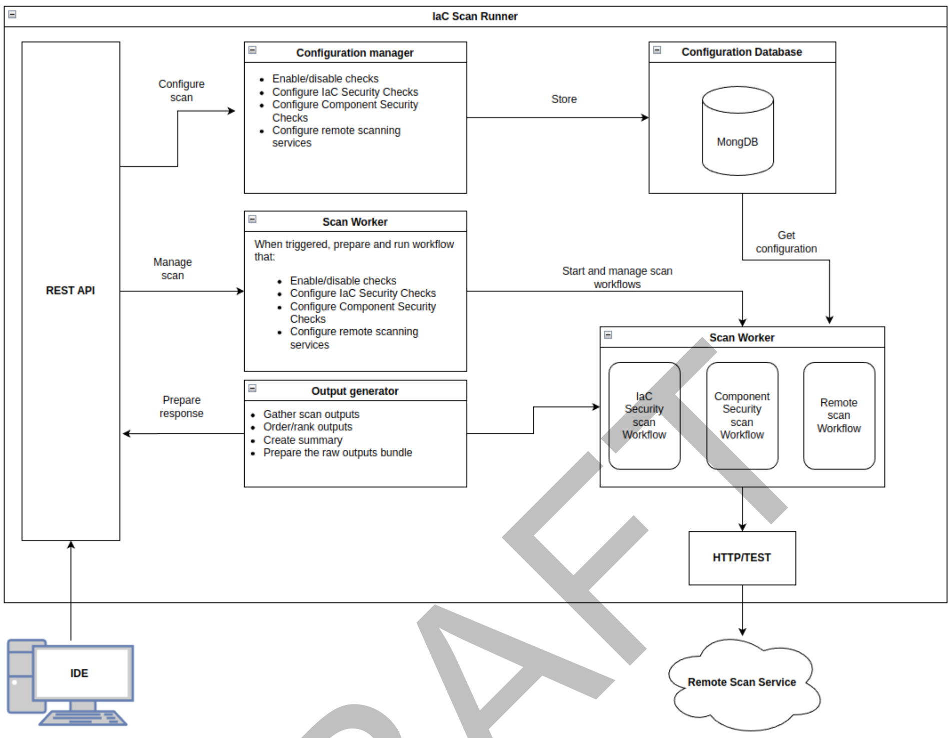

IaC Scan Runner
REST API
Configure scan
Configuration manager
Enable/disable checks
Configure IaC Security Checks
Configure Component Security Checks
Configure remote scanning services
Store
Configuration Database
MongDB
Get configuration
Manage scan
Scan Worker
When triggered, prepare and run workflow that:
Enable/disable checks
Configure IaC Security Checks
Configure Component Security Checks
Configure remote scanning services
Start and manage scan workflows
Scan Worker
IaC Security scan Workflow
Component Security scan Workflow
Remote scan Workflow
Prepare response
Output generator
Gather scan outputs
Order/rank outputs
Create summary
Prepare the raw outputs bundle
HTTP/TEST
IDE
Remote Scan Service

Figure 4: Component diagram

1.2.2 Components description

The IaC Scan Runner (Figure 4) is comprised of the following components:

- REST API is the main interface that will be called by the PIACERE IDE. In particular cases, if needed it would be also possible to create a corresponding CLI application that could make the call to the API, which would enable integration of the tool in the continuous integration (CI) scripts.
- The configuration manager takes care of the IaC Scan Runner project configurations. Each project includes the set of available checks, list of enabled/disabled checks for scan, configuration of each integrated or remote check. For example, remote scan services will need the URL and credentials to perform the scans.
- Configuration database stores all project settings of the installed checks and provide them when some scans are performed.
- The Scan Worker takes care of scan execution. This means that it takes all checks and configurations of the same type (IaC, Component or Remote) and prepares workflows to be executed. The output of the scans is collected by the output generator.
- Scan workflows presents a set of processes that perform scans.
- The output generator gathers the outputs of scans and forms the output for IDE. This includes filtering outputs, creating the summary, and ordering and ranking the outputs.

All components are designed to run together inside a docker container, which can be set up locally on a developer's machine or be available as a service.

1.2.3 Technical specifications

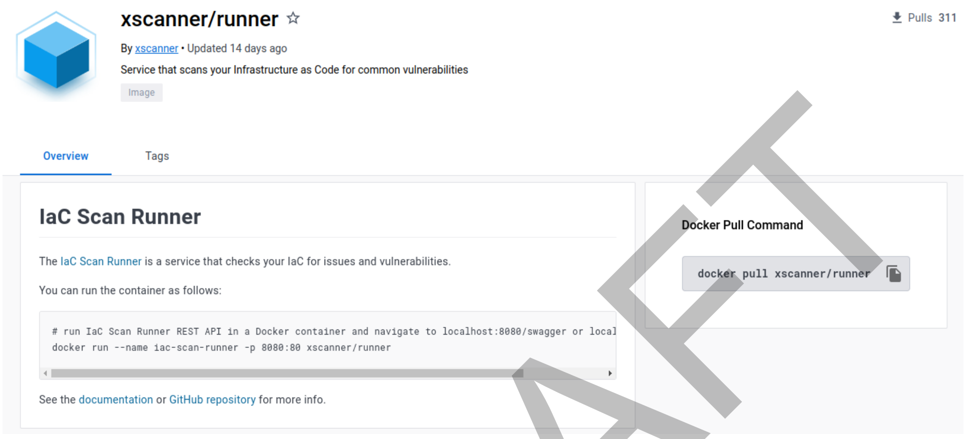
The IaC Scan Runner that is developed within PIACERE is written in Python programming language. The REST API uses OpenAPI Specification, whereas Swagger UI [7] and ReDoc are used to document it. The general documentation (in Figure 6) for IaC Scan Runner uses Sphinx documentation tool (with Read the Docs theme), where the docs can be easily rendered from RST files. The IaC Scan Runner CLI, which is used to run the REST API from the console is also written in Python and is regularly published on Python Package Index (PyPI [8]) as the *iac-scan-runner* pip package (the development version of the package is available on Test PyPI). The API is designed using FastAPI - a modern and high-performance web framework and the CLI is build using Click-based Typer Python library. Apart from a local installation, both REST API and docs can be also distributed as a Docker image, where the images are stored within the *xscanner* Docker Hub community organization. The Docker image for the IaC Scan Runner tries to be as general as possible (currently it is based on Python slim-buster Debian release). This allows installing almost all possible checks, because the check linters and tools are usually written in all different languages and therefore require different installation procedures. The download and installation of checks is initiated by a simple bash script, which ensures that the checks are available to be used by the REST API.

2 Delivery and usage

This section will first describe the package info and the installation of IaC Scan Runner and then its usage through the REST API and CLI.

2.1 Package information

The IaC Scan Runner module is delivered as a Docker application including a service accessible through an API. The *xscanner/runner* [9] Docker image (Figure 5) is updated and published regularly on Docker Hub. The CLI that is currently able to run the API is available as *iac-scan-runner* Python package and is published on PyPI [8] . Both, API and CLI use semantic versioning for new releases and the latest available version is 0.3.0.



xscanner/runner

By xscanner • Updated 14 days ago

Service that scans your Infrastructure as Code for common vulnerabilities

IaC Scan Runner

The IaC Scan Runner is a service that checks your IaC for issues and vulnerabilities.

You can run the container as follows:

```
# run IaC Scan Runner REST API in a Docker container and navigate to localhost:8080/swagger or local
docker run --name iac-scan-runner -p 8080:80 xscanner/runner
```

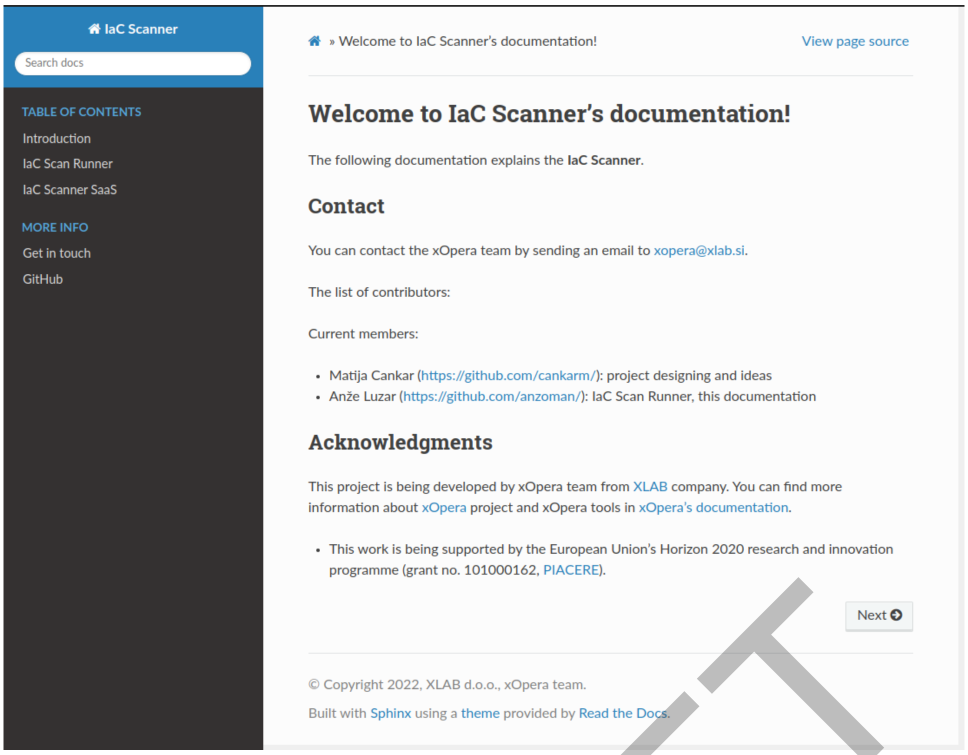
See the documentation or GitHub repository for more info.

```
docker pull xscanner/runner
```

Figure 5: Docker Image

Some of the IaC Scan Runner services are already available to the PIACERE consortium partners on public links:

- REST API: https://scanner.xopera.piacere.esilab.org/iac-scan-runner/
- Swagger UI: https://scanner.xopera.piacere.esilab.org/iac-scan-runner/swagger/
- ReDoc: https://scanner.xopera.piacere.esilab.org/iac-scan-runner/redoc/
- Documentation: https://scanner.xopera.piacere.esilab.org/docs/



Welcome to IaC Scanner's documentation!

The following documentation explains the **IaC Scanner**.

Contact

You can contact the xOpera team by sending an email to xopera@xlab.si.

The list of contributors:

Current members:

- Matija Cankar (https://github.com/cankarm/): project designing and ideas
- Anže Luzar (https://github.com/anzoman/): IaC Scan Runner, this documentation

Acknowledgments

This project is being developed by xOpera team from XLAB company. You can find more information about xOpera project and xOpera tools in xOpera's documentation.

- This work is being supported by the European Union's Horizon 2020 research and innovation programme (grant no. 101000162, PIACERE).

© Copyright 2022, XLAB d.o.o., xOpera team.

Built with Sphinx using a theme provided by Read the Docs.

Figure 6: Documentation page

2.2 Installation instructions

This section will cover the various installation options available for the IaC Scan runner. The installation process will differ depending on how the user intends to run the IaC REST API. MongoDB is essential as it stores the project configurations used by the IaC scan runner. If user did not use Docker Compose for setup, please execute following commands:

```
# Export env variables
export MONGODB_CONNECTION_STRING=mongodb://localhost:27017
export SCAN_PERSISTENCE=enabled
export USER_MANAGEMENT=enabled

# Setup MongoDB
$ docker run --name mongodb -p 27017:27017 mongo
```

2.2.1 Running with Docker

The REST API can be run using a public xscanner/runner Docker image as follows:

```
# run IaC Scan Runner REST API in a Docker container and navigate to
localhost:8080/swagger or localhost:8080/redoc
$ docker run --name iac-scan-runner -p 8080:80 xscanner/runner
```

It is also possible to build the image locally. To do so follow these steps:

```
# Clone the repository
$ git clone https://github.com/xlab-si/iac-scan-runner.git
```

```
# Move to the repository and run following command
$ docker build -t iac-scan-runner .

# Now image can me run by following command
$ docker run --name iac-scan-runner -p 8080:80 iac-scan-runner
```

2.2.2 Run from CLI

To run using IaC Scan Runner CLI execute following commands:

```
# Install the CLI
$ python3 -m venv .venv && . .venv/bin/activate
(.venv) $ pip install iac-scan-runner

# print OpenAPI specification
(.venv) $ iac-scan-runner openapi

# Install prerequisites
(.venv) $ iac-scan-runner install

# Run IaC Scan Runner REST API
(.venv) $ iac-scan-runner run
```

2.2.3 Run from source

To run locally from source with uvicorn:

```
# Clone the repository
$ git clone https://github.com/xlab-si/iac-scan-runner.git

# Move to cloned repository and install prerequisites
$ python3 -m venv .venv && . .venv/bin/activate
(.venv) $ pip install -r requirements.txt
(.venv) $ ./install-checks.sh


# run IaC Scan Runner REST API (add --reload flag to apply code changes on the way)
(.venv) $ cd src
(.venv) $ uvicorn iac_scan_runner.api:app
```

To run locally from source with docker compose:

```
# Clone the repository
$ git clone https://github.com/xlab-si/iac-scan-runner.git

# Move to cloned repository and run following command
docker compose up
```

2.3 User Manual

The IaC Scan Runner API enables users to interact with the primary IaC inspection component and start IaC scans. It offers a range of IaC checks that can be filtered and customized to suit specific requirements. Users can select either all or a subset of checks to be executed during an IaC scan. Upon completion of the scan, the API will provide a comprehensive report of all the check results.

The default endpoints in the API are feature that has been deprecated, meaning they are no longer recommended for use because they may be removed in the future. However, some back compatibility endpoints still exist and are listed in Table 3.

Table 3: Back-compatibility API calls

REST API Endpoint	Description
GET /default/checks	Retrieve and filter supported checks
PUT /default/checks/{check_name}/enable	Enable check for running
PUT /default/checks/{check_name}/disable	Disable check for running
PUT /default/checks/{check_name}/configure	Configure check
POST /default/scan	Initiate IaC scan

Project endpoints allow user to set up and configure check on a per-project basis. These endpoints, presented in Table 4, provide functionality for creating, editing projects as well as configuring checks.

Table 4: IaC Scan Project management API calls.

REST API Endpoint	Description
GET /project/results	Get scan results per project
GET /project/checks	Retrieve and filter supported checks per project
PUT /project/checks/{check_name}/enable	Enable check for running per project
PUT /project/checks/{check_name}/disable	Disable check for running per project
PUT /project/checks/{check_name}/configure	Configure check per project
POST /project	Generate new scan project
POST /project/scan	Initiate IaC scan
DELETE /project/results/{uuid}	Delete scan result per result uuid

If your IaC Scan Runner is already set up you can navigate to /swagger or /redoc where you can user the interface to test the API. In this example, we will use curl for calling the API endpoints.

1. Lets create a project named test.

```
curl                      -X                      'POST'                   \
  'http://0.0.0.0/project?creator_id=test'                                 \
  -H               'accept:               application/json'                \
  -d ''
```

2. For example, lets say that we want to initaite all check expect ansible-lint. Lets disable it.

```
curl                         -X                       'GET'                 \
'http://0.0.0.0/project/checks?project_id=1e7b2a91-2896-40fd-8d53-
83db56088026'                                                               \
  -H 'accept: application/json'
```

3. For IaC-Scan-Runner to work files are expected to be a compressed archives (usually zip files). In this case response type will be json, but it is possible to change it to html. Please change YOUR.zip to path of your file.

```
curl -X 'POST' \
'http://0.0.0.0/project/scan?project_id=1e7b2a91-2896-40fd-8d53-
83db56088026&scan_response_type=json' \
-H 'accept: application/json' \
-H 'Content-Type: multipart/form-data' \
-F 'checks=' \
-F 'iac=@YOUR.zip;type=application/zip'
```

2.4 Licensing information

IaC Scan Runner is licensed under open-source **Apache License 2.0**.

2.5 Download

The source code for IaC Scan Runner is available within xlab-si/iac-scan-runner GitHub repository and the documentation is visible in xlab-si/iac-scanner-docs or can be explored publicly on GitHub Pages. GitHub Actions are being used for the CI/CD tests and for building Docker images and packages.