Persistent

A tool-first, automated approach to scaling your DevSecOps organization

Leveraging ExtenSURE — Persistent's Intelligent Product Engineering Framework

Changes in current development environments and the importance of DevSecOps

Software Development Lifecycle has completely changed over the last few years. The adoption of agile project methodology has become increasingly critical to deliver products faster and reduce time to market. This needs to be done with the best code quality, zero security vulnerabilities, and the ability to scale as the usage of the application increases.

All this complexity has necessitated the need to adopt tools, technologies, and practices that can cover the breadth of software development lifecycle (SDLC) and offer automated validations, testing, and remediation wherever possible.

DevSecOps or Secured SDLC (SSDLC) is a fundamental shift in how organizations manage software engineering. It is driven by the fact that the business needs to move rapidly to develop new software faster for customers which is resilient, agile, and free of any vulnerabilities or license risks.

Despite the notable progress we've witnessed many organizations still struggle to move beyond the middle stages of DevOps. They can struggle to scale their DevOps methods of working beyond development, operations and security teams to evolve into a scaled DevSecOps organization.

But some organizations do succeed. They expand DevSecOps practices beyond the initial early-adopting teams, continuing to evolve and improve across the organization. What makes the difference?

Scaling DevSecOps practices

Applying DevSecOps at scale requires developing the process to support the organization in releasing features on a more frequent basis at the program / portfolio level. Before applying these principles, leadership must ensure the team has a solid foundation and the following fundamentals are in place:

1

Code quality

Clean architecture which enables scrum teams to work efficiently and resolve issues early.

2

Ability to build in parallel

The ability to develop and manage different artifacts as independent components.

3

A high degree of automation

Automated code reviews, test automation, and deployment.

Measuring the scale and impact of your DevSecOps using DORA metrics

Everyone who uses DevSecOps is aware of the DORA metrics created through six years' worth of surveys conducted by DORA (DevSecOps Research and Assessments).

Throughput

Measured using deployment frequency (DF), lead time for changes (MLT)

Stability

Measured using the time to restore service (MTTR), change failure rate (CFR).

Using these metrics, development teams are categorized into elite, high, medium, and low. While measuring these metrics is essential, what is critical is to know the value and benefits of doing things as part of software delivery which helps us become elite performers.

Scale DevSecOps using ExtenSURE — Persistent's Intelligent Product Engineering framework

ExtenSURE uses a tool-first, engineering-driven approach to scale DevSecOps. It enables teams to onboard applications, run them through an automated CI / CD pipeline, and significantly improve efficiencies by:

1
Increasing speed of deployment.

2
Improving software stability and lowering change failure rates.

3
Building security in from the start and spending less time fixing security issues.

4
Monitoring errors in each deployable pre-production environment.

5
Detecting production errors and assessing the impact in real-time.

The key tenets of ExtenSURE's approach

Baselining and enhancing code quality:

- Code complexity and hotspots
- Code contributors and team dynamics
- Code structure and dependencies
- Reverse-engineering architecture
- Fitness tests for architecture patterns

Defect prevention over defect detection:

- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Software Composition Analysis (SCA)
- Infrastructure-as-a-Code Security Analysis

Intelligent operations:

- Resiliency and Observability using 'Chaos Engineering'
- Production Application Monitoring (APM)
- User Experience Monitoring

Automated deployment pipeline

- Automated Environment and Pipeline Provisioning

Baselining and enhancing code quality

For an application that has been developed over many years, runs into millions of lines of code, and has hundreds of developers who contributed, it becomes important to have an excellent understanding of the code based on the evolution of the code. This gives the ability to predict its behavior and find hotspots that are prone to defects. This baselining can futureproof the codebase and prevent bottlenecks and maintenance issues.

The ExtenSURE framework begins with an in-depth analysis of the codebase using automated product analysis and observability tools. The analysis covers application architecture, static code and data analysis, code coverage, code complexity, log analysis, documentation, bugs, and security vulnerabilities. Based on this analysis, key vectors such as complexity and churn scores can be calculated for various code modules of the product codebase. These vectors help generate actionable insights to plan and prioritize activities for improving the quality of the codebase.

Code complexity and hotspots

- Assesses current technical debt.
- Identifies and prioritizes critical hotspots in code, based on previously deployed fixes.
- Identifies and prioritizes complex modules.

Code contributors and team dynamics

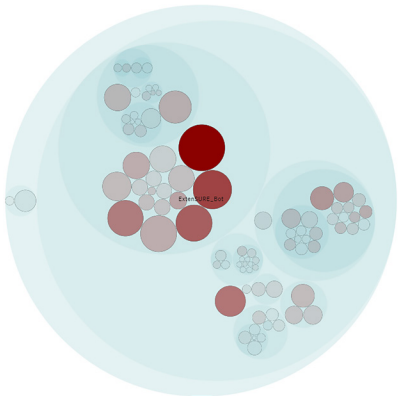
- Identifies top code contributors.
- Helps understand team dynamics and dependencies.
- Accurately measures flight risk by combining knowledge risk view analysis with an individual knowledge map.

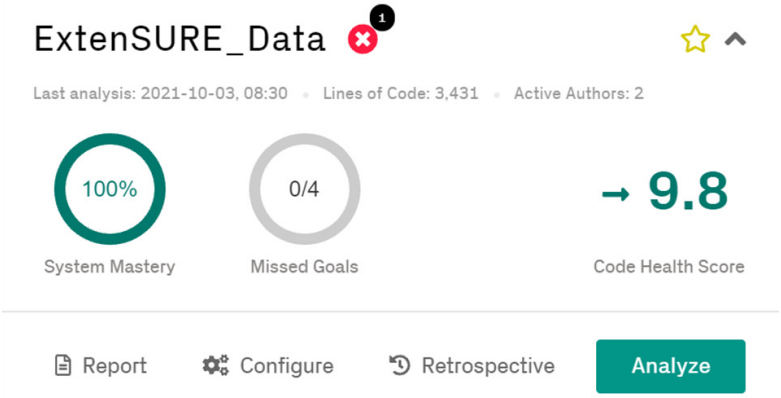
Code structure and dependencies

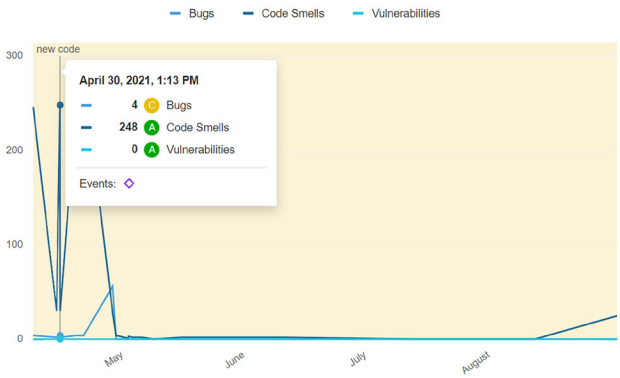
- Helps identify dependencies across various modules to understand the structure of the product codebase and reverse engineer architecture and design.
- Automatically generates technical documentation for unknown / non-documented databases.
- Extracts business rules from application codebase to understand and document functionality.

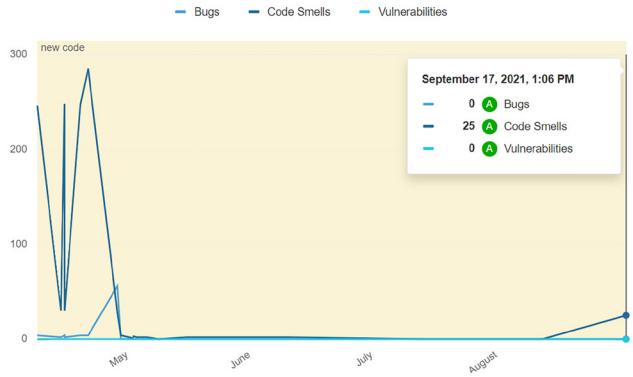
Why ExtenSURE's codebase analysis phase is a crucial first step

It helps prioritize and focus on crucial code modules to ensure the engineering team does not just fix superficial bugs but looks deeper into the issues and tackles them at the root.

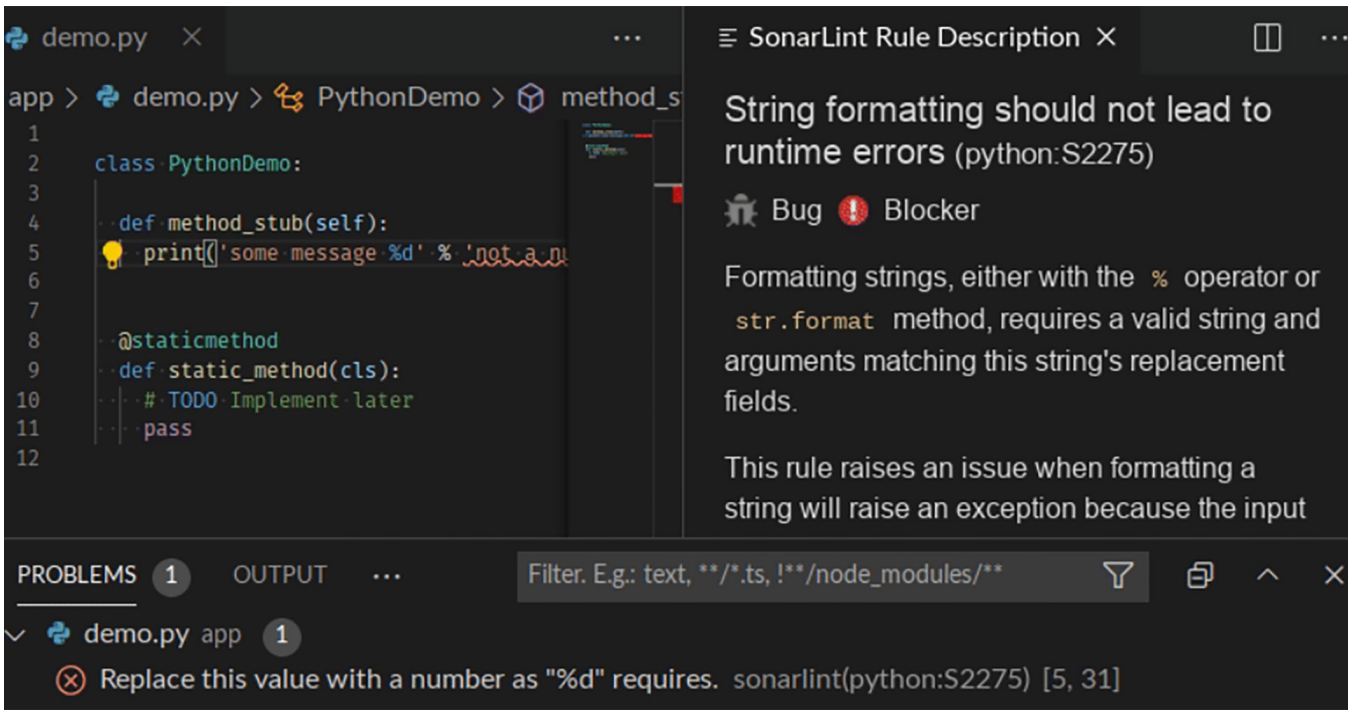








Analyzing the code changes when merging code is an excellent idea. However, it adds an extra step before the developer is aware of the issues in their code. To enable developers to truly own their code, these issues need to be highlighted at the code writing stage itself. ExtenSURE leverages IDE extensions, enforcing code hygiene rules in the code quality gate analysis.



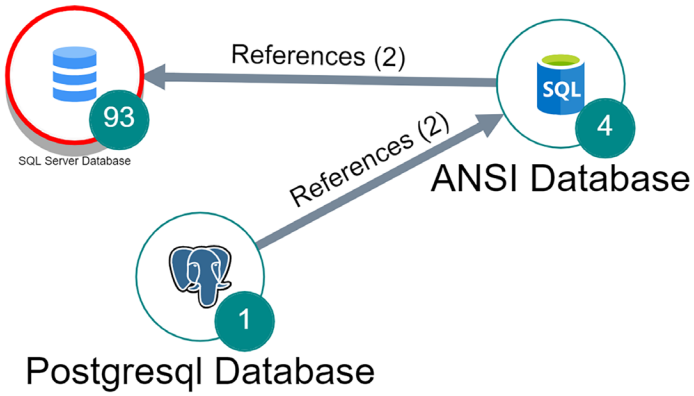
Baselining and improving overall codebase health is the first step towards DevSecOps transformation, and this crucial first step is aimed at achieving that through:

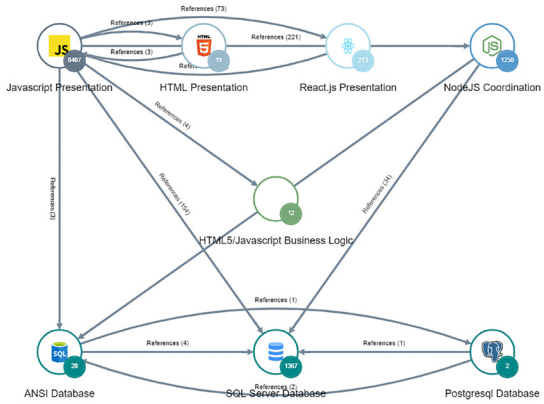
- **Automated code reviewed:** Use analysis results from the tool as early feedback on code.
- **Issue prioritization:** Prioritize code that is changing most frequently.
- **Developer sensitization:** Educate developers towards better coding practices.
- **Consistent quality standard:** Enforce consistent quality standards across teams.
- **Visibility:** Provide visibility to everyone in the team through standard, quality metrics.

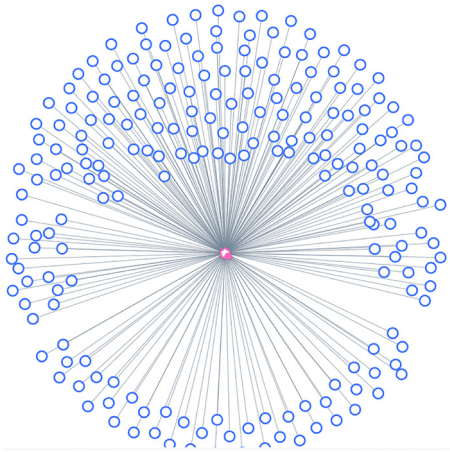
Reverse-engineering architecture

Consider a portfolio of applications that have been developed over many years by several developers using different technology stacks. Many of these developers who wrote the code would have either moved on and will not be available with the current team.

The ExtenSURE framework provides the ability to auto reverse-engineer all database structures, code components, and interdependencies in complex software systems down to the tiniest details and creates accurate, interactive architecture blueprints. Individual developers can see how their changes impact the architecture and indicate the changes to the affected team.







Fitness tests for architecture patterns

Fitness functions describe how close an architecture is to achieving an architectural aim. Regardless of the application architecture (monolith, microservices, or other), fitness function-driven development can introduce continuous feedback for architectural conformance and inform the development process as it happens, rather than after the fact.

ExtenSURE leverages fitness testing for an architecture patterns approach and integrates this with unit testing using libraries like ArchUnit (Java) and NetArchTest (.NET) for automatically testing architecture and coding rules. It also checks dependencies between packages and classes, class design, naming and dependency, and more.

TestCase Name	Status	Exception
BasicTest.NamingConvention.controller_should_be_suffixed	**Fail**	Architecture Violation [Priority: Medium] – Rule classes ta=hat reside in a package '...controller...' should have simple name ending with 'Controller" was violated (1 times): simple name of demo.controller.demo does not end with 'Controller' in (demo.java.0)
BasicTests.NamingConvention.interfaces_with_repository_should_be_in_a_repository_package	**Pass**	
BasicTests.NamingConvention.service_should_be_infix_with_service_word	**Pass**	
BasicTests.CycleCheck.cycle_check	**Fail**	Architecture Violation [Priority: MEDIUM] – Rule 'slices matching '...demo...(*)...' should be free of cycles' was violated (1 time): Cycle detected: Slice firstservice -> Slice secondservice -> Slice firstservice 1. Dependencies of Slice firstservice – Field <demo.service.firstservice.Firstservice.obj> has type <demo.service.secondservice.Secondservice> in (FirstService.java:0) 2. Dependencies of Slice secondservice - Field <demo.service.secondservice.SecondService.obj> has type <demo.service.firstservice.FirstService> in (SecondService.java:0)

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Suite						
CycleCheck	0	0	1	69		
DependencyCheck	1	0	0	27		
LayerAcess	1	0	0	11		
NamingConvention	2	0	1	8		
LocationChecks	3	0	1	18		
Total	**7**	**0**	**3**	**133**		

Class	Method	Start	Time (ms)
Suite			
CycleCheck — Failed			
BasicTests.CycleCheck	cycle_check	1625806644651	57
DependencyCheck — Passed			
BasicTests.DependencyCheck	entity_class_should_not_depend_on_anyone	1625806644760	26
LayerAcess — Passed			
BasicTests.LayerArchitecture	layer_architecture	1625806644806	9
NamingConvention — Failed			
BasicTests.NamingConvention	controllers_should_be_suffixed	1625806644836	1
NamingConvention — Passed			
BasicTests.NamingConvention	interface_with_repository_should_be_in_a_repository_package	1625806644838	2
	service_should_be_infix_with_service_word	1625806644840	2

Defect Prevention over Defect Detection

Software engineering teams must maintain a testing mindset and rigor as early as the design phase, through deployment and even beyond to production — monitoring production data and logs and creating a feedback loop to engineering teams.

An automated 'Shift Left' approach to quality engineering focuses on reliability, security, and accessibility at all stages of the design, development, and deployment process. It ensures accelerated software releases and predictability in product releases by eliminating nasty surprises and helping to prevent cost overruns.

This approach primarily requires an automation-first approach by leveraging our AI-ML expertise coupled with best-in-class tools for a strong testing mindset across every stage in the software product engineering lifecycle.

The ExtenSURE framework introduces comprehensive testing as early as the design phase and continues post-deployment.

Static Application Security Testing (SAST)

ExtenSURE, as part of its CI-CD pipeline, leverages SonarQube and CodeScene to do a comprehensive validation of the code against security vulnerabilities, identify hotspots, and code health for these hotspots, look at team dynamics, and provide remediation measures.

Before

After

Dynamic Application Security Testing (DAST)

The framework enforces robust DAST during the integration testing phase by leveraging OWASP ZAP (Zed Attack Proxy) to highlight low, medium, and high risks with suggestions to fix them.

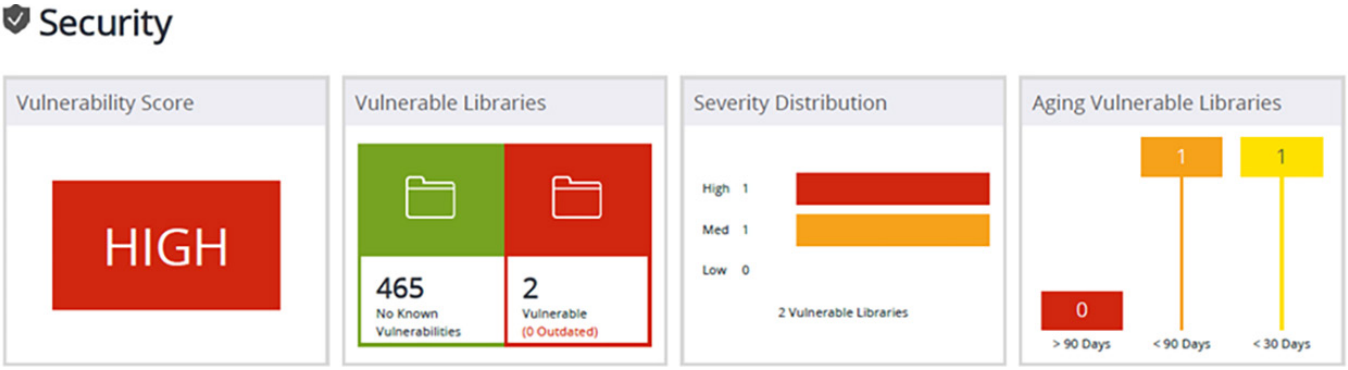
Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	12
Cross-Domain Misconfiguration	Medium	52
Source Code Disclosure — Java	Medium	19
Vulnerable JS Library	Medium	1
X-Frame-Options Header Not Set	Medium	10
Dangerous JS Functions	Low	4
Permissions Policy Header Not Set	Low	34
Private IP Disclosure	Low	1
Server Leaks Version Information via “Server” HTTP Response Header Field	Low	56
X-Content-Type-Options Header Missing	Low	50

Examples of the alerts that provide valuable insights for the development team to look at and fix before the application gets deployed to production.

SAST should be performed early and often against all files containing source code. DAST should be performed on a running application in an environment like production. The best approach is to include both SAST and DAST in your application security testing program.

Software Composition Analysis (SCA)

The extensive use of open-source libraries and Docker images brings added risks of open library vulnerabilities, license compliance risks (MIT, Apache, GPL), and security vulnerabilities. The framework leverages WhiteSource, CAST highlights to perform detailed analysis, fixing issues early during software development.



Infrastructure-as-a-Code Security Analysis

The National Security Agency (NSA) and the Cybersecurity and Infrastructure Security Agency (CISA) released a cybersecurity technical report, “Kubernetes Hardening Guidance,” in August 2021. This report details threats to Kubernetes environments and provides configuration guidance to minimize risk.

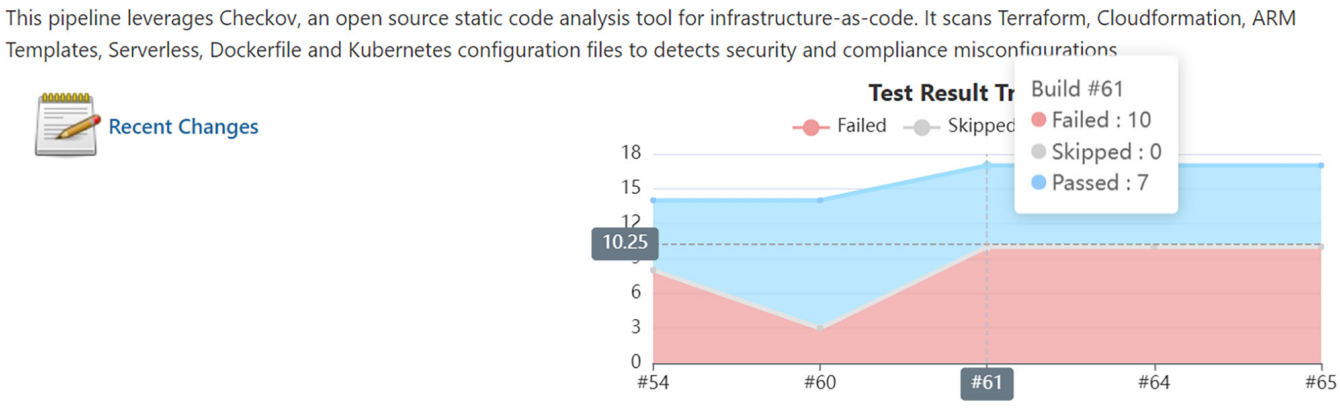
The framework recognizes the role of IaaC in securing cloud-native environments early on and delegates security ownership to individual contributors while also distributing it across existing frameworks within CI/CD pipelines.

The ExtenSURE framework shifts security feedback further left by integrating Checkov and other relevant tools. Checkov includes over 200 new policies and a Docker file scanner that help ensure container images are built securely, without misconfigurations.

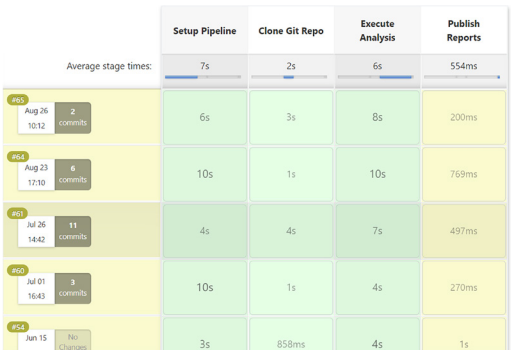
Some of the tests which must be performed in accordance with NSA and CISA guidance are:

Non-root containers and rootless container engines.	Immutable container filesystem.	Privileged containershost PIDD, hostIPC privileges.	hostNetwork access.	Ingress and Egress blocked.

Pipeline checkov-analysis



Stage View



All Failed Tests

Test Name
checkov.dockerfile.checks.HealthcheckExists.dockerfile CKV_DOCKER_2/Ensure that HEALTHCHECK instructions have been added to contai
checkov.dockerfile.checks.HealthcheckExists.dockerfile CKV_DOCKER_2/Ensure that HEALTHCHECK instructions have been added to contai
checkov.dockerfile.checks.HealthcheckExists.dockerfile CKV_DOCKER_2/Ensure that HEALTHCHECK instructions have been added to contai
checkov.dockerfile.checks.HealthcheckExists.dockerfile CKV_DOCKER_2/Ensure that HEALTHCHECK instructions have been added to contai
checkov.dockerfile.checks.HealthcheckExists.dockerfile CKV_DOCKER_2/Ensure that HEALTHCHECK instructions have been added to contai
checkov.dockerfile.checks.RootUser.dockerfile CKV_DOCKER_8/Ensure the last USER is not root /jenkins/Dockerfile.USER
checkov.dockerfile.checks.UserExists.dockerfile CKV_DOCKER_3/Ensure that a user for the container has been created /codescene/Dockerfi
checkov.dockerfile.checks.UserExists.dockerfile CKV_DOCKER_3/Ensure that a user for the container has been created /mongo/Dockerfile.

CONTROL NAME	FAILED RESOURCES	WARNING RESOURCES	ALL RESOURCES	% SUCCESS
Allow privilege escalation	0	0	3	100%
Allowed hostPath	1	0	3	66%
Applications credentials in configuration files	0	0	3	100%
Automatic mapping of service account	0	0	0	NaN
CVE-2021-25741 - Using symlink for arbitrary host file system access.	0	0	3	100%
Cluster-admin binding	0	0	0	NaN
Container hostPort	0	0	3	100%
Control plane hardening	0	0	3	100%
Dangerous capabilities	0	0	3	100%
Exec into container	0	0	0	NaN
Exposed dashboard	0	0	1	100%
Host PID/IPC privileges	1	0	3	66%
Immutable container filesystem	2	0	3	33%
Ingress and Egress blocked	2	0	3	33%
Insecure capabilities	0	0	3	100%
Linux hardening	2	0	3	33%
Network policies	0	0	0	NaN
Non-root containers	0	0	3	100%
Privileged container	1	0	3	66%
Resource policies	0	0	3	100%
hostNetwork access	1	0	3	66%
21	10	0	49	79%

Intelligent Operations

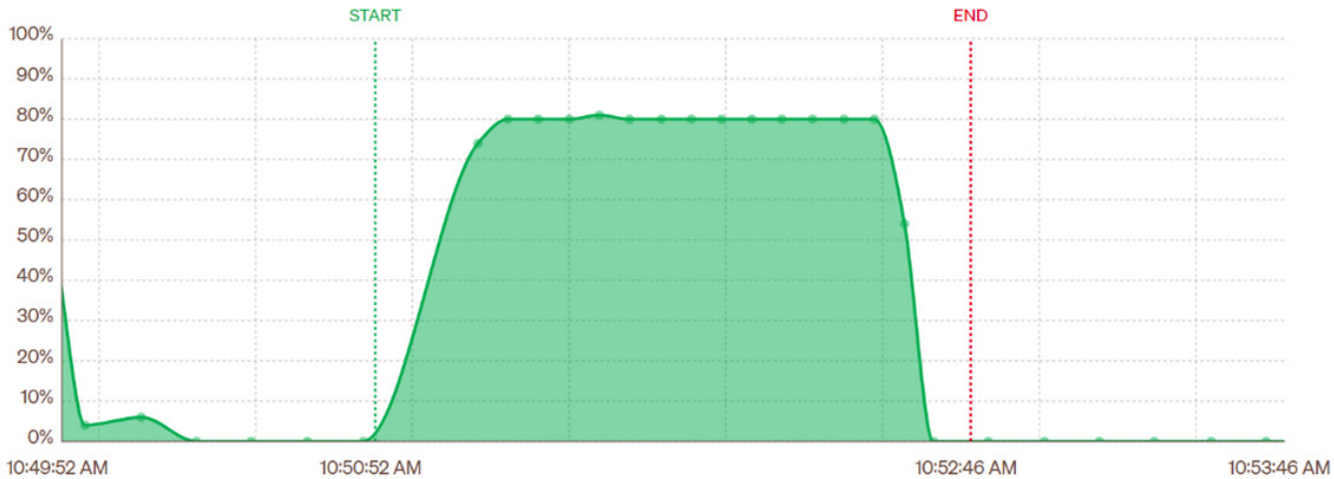
Resiliency and Observability using 'Chaos Engineering'

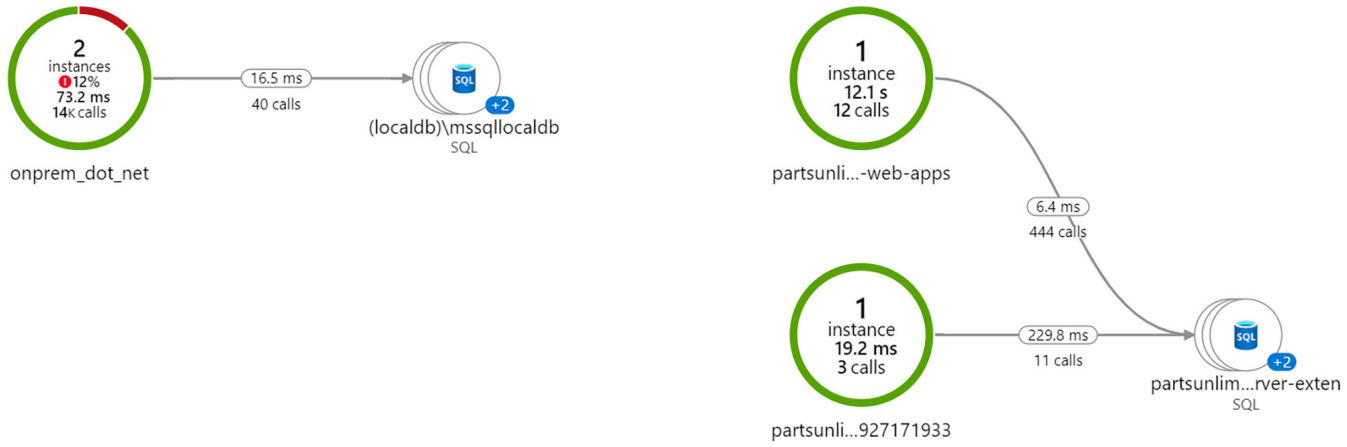
The framework uses principles of chaos engineering to stress-test applications and infrastructure for unknown unknowns. It simulates extreme failures and edge cases, preparing for issues under scenarios such as:

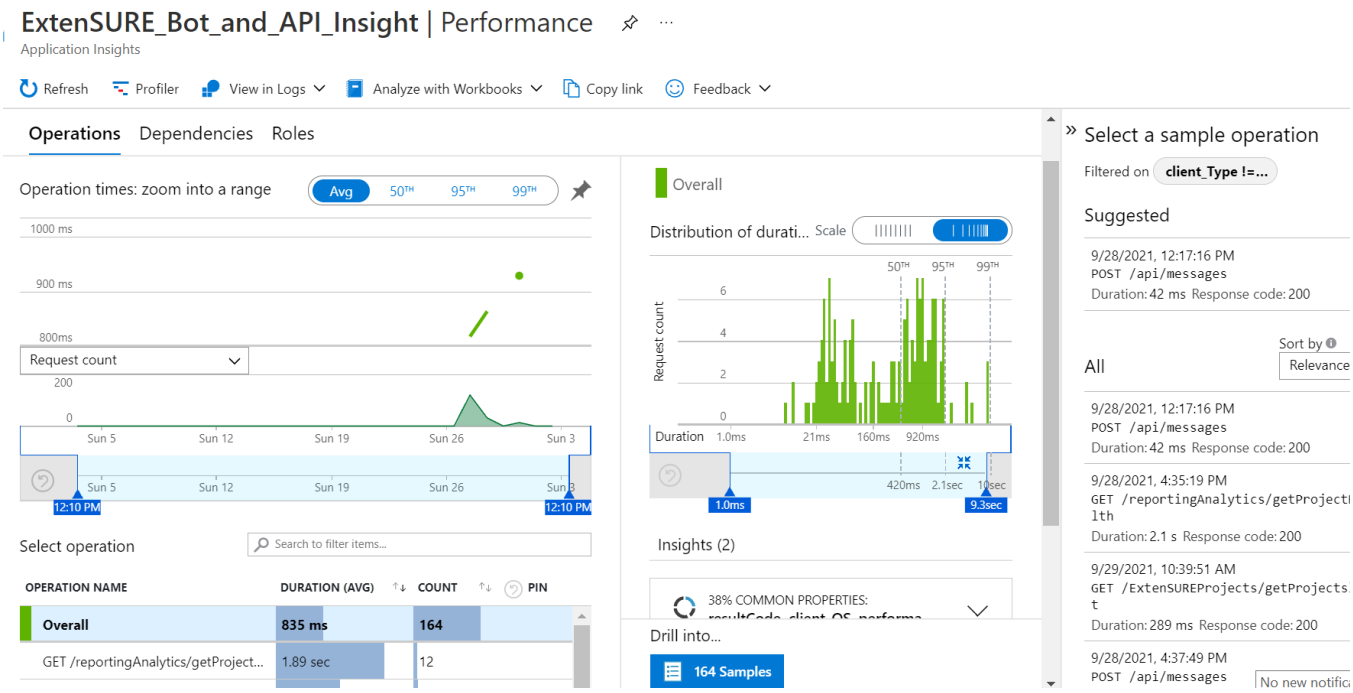
- Application behavior when the CPU utilization reaches 100% — Does it auto-scale or crash?
- Application behavior when 30 – 40% packets drop for some microservices — Slowness observed and possible impact on NPS.
- Application behavior when a critical infrastructure pod is suddenly shut down.

Number of Healthy Hosts

CPU Utilization







Observing how systems react to such failures allows teams to find ways to improve their resilience around metrics such as:

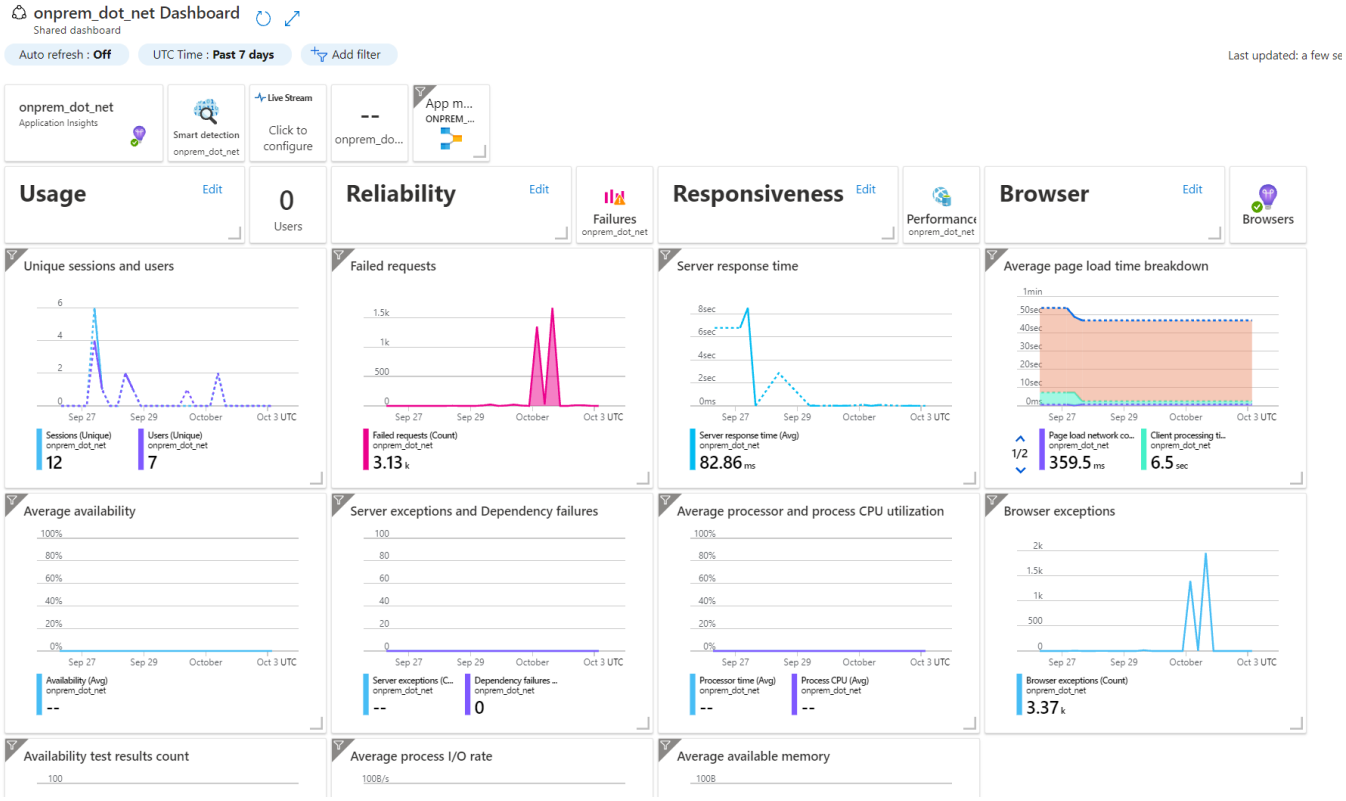
Uptime: Lost revenue and added costs.

Meantime between failures (MTBF): Customers attrition.

Meantime to resolution (MTTR): Fixing defects or developing new functionalities.

Production Application Monitoring (APM)

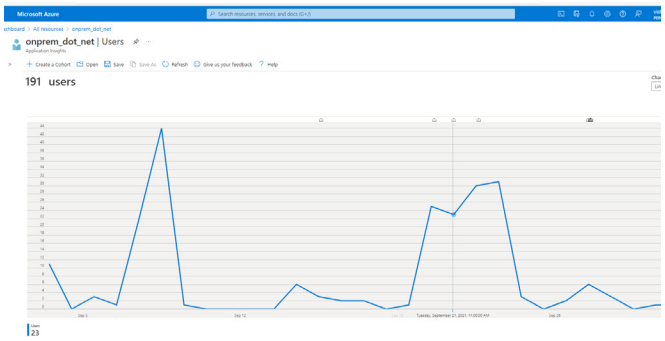
The framework prioritizes monitoring the production application by leveraging Azure Application Insights to monitor production applications 24/7 for security issues and API failures to ensure robust application performance after deployment.

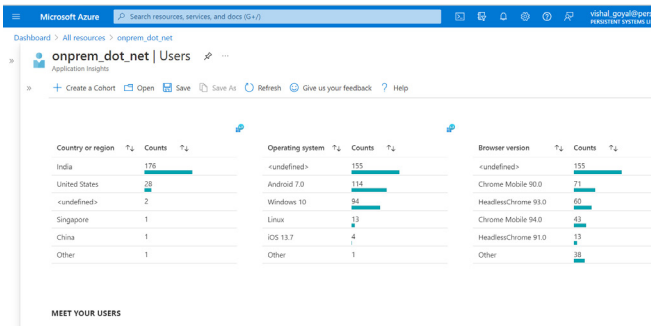


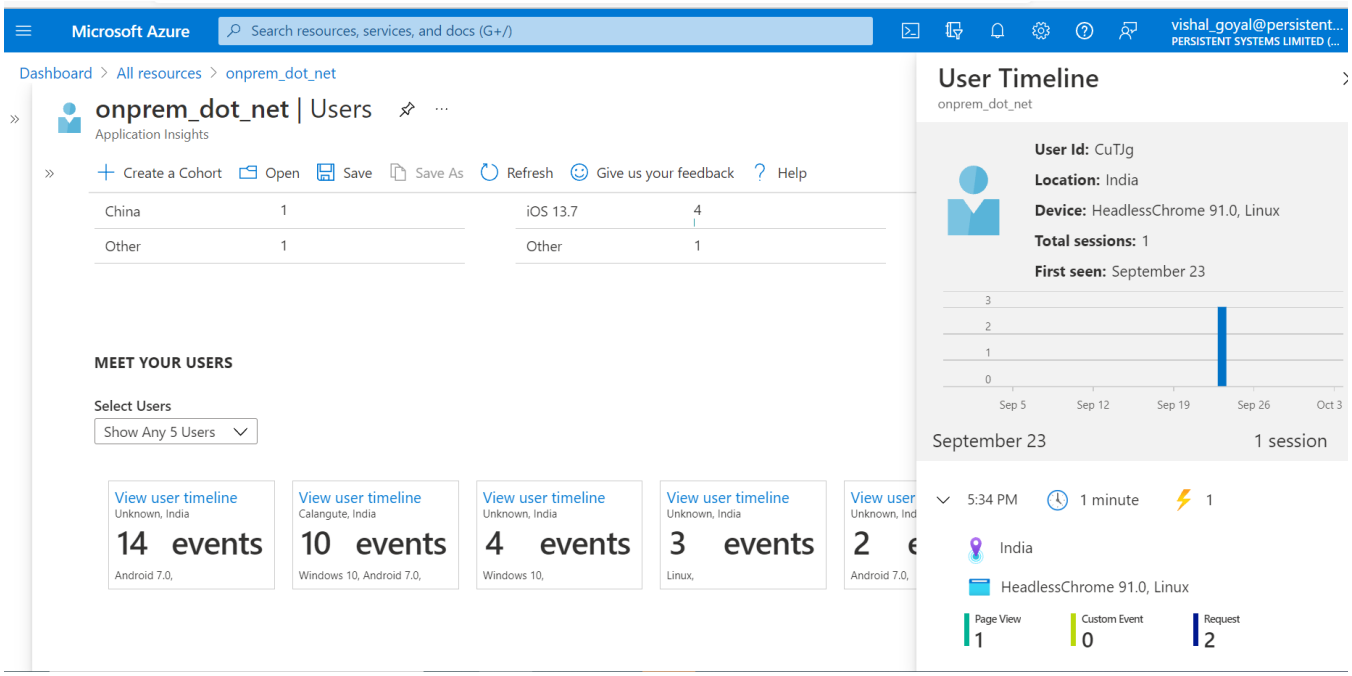
User Experience Monitoring

Beyond monitoring the application's performance, it is also essential to observe how an application is being used and navigated through, and which channels are being used more in omnichannel applications. This information helps improve the user experience by continuing to enhance the application as per the analysis.

The ExtenSURE framework leverages Azure Application Insights to instrument the production application telemetry and monitors the user analytics.







Automated Deployment Pipeline

Automated Environment and Pipeline Provisioning

Environments would be required to execute a large number of automated test cases on an ongoing basis and need to be as much like production as possible to detect issues early. Consider the following when provisioning the environment:

1\ The orchestrated installation of software, parallel installation, and handling dependent applications.

2\ Tools for monitoring the infrastructure and application, including the log and event collection.

3\ Environment creation needs to be automated and version-controlled. Scripted environments approach would create a script that can be executed on a server or virtual machine to install / configure the system. Configuration management tools like Chef and Puppet would help in creating the scripts and storing them under source control.

4\ Consider having simulators and emulators which help increase test coverage and provide feedback to resolve any defects early.

5\ Containers provide a way for applications to be portable and easily managed.

ExtenSURE leverages a software engineering mindset to build multiple automation blocks using infrastructure-as-code, pipeline-as-code, and configuration-as-code to stitch the e2e approach by leveraging dockers, ansible and terraform templates.

This allows software engineers to request and provision resources on-demand without system administrators doing this manually, significantly helping scale and speed up operations across different environments. This also helps eliminate dependencies and bottlenecks for a software development team waiting for resources from a separate system administration team.

\ IaaC solution drastically reduces the manual efforts (> 90%) by efficiently and consistently provisioning on-demand, pre-configured dev/test environment blueprints.

\ Enables CI / CD ramp-up exponentially across project teams. The overall time to setup and provide a containerized tooling stack of opensource offerings is achieved in a matter of minutes versus a manual approach taking days.

\ The 1-click environment automation pipeline solution is extensible by customizing the DevOps cartridges to the needs of a given technology blueprint (both cloud and on-prem).

Conclusion

With the changing dynamics of software development, it's critical to implement a DevSecOps pipeline that can address multiple business needs without burdening the engineering teams.

- Adopt a shift left approach for your SDLC.
- Ensure that the development pipeline looks at resiliency, observability, and security as critical elements rather than afterthoughts. The cost of remediating any kind of defects or issues in post-production is far higher compared to addressing it in the earlier stages of the SDLC.
- Adopt practices that help do architecture engineering and analysis of the source code for better knowledge management, which will also help reduce technical debt and keep code quality higher.
- Look at the evolution of the code and not just static code for better team management and prioritizing the technical debt.
- Automate whatever is possible: code quality checks, security, infrastructure deployment, pipeline deployment.

About Persistent

With over 14,500 employees located in 18 countries, Persistent Systems is a global services and solutions company delivering Digital Engineering and Enterprise Modernization. We combine deep technical expertise and industry experience to help our clients anticipate what's next and develop solutions that create unique competitive advantage. Persistent was named to the Forbes Asia Best Under a Billion 2021 list, representing consistent top-and bottom-line performance as well as growth.

India
Persistent Systems Limited
Bhageerath, 402
Senapati Bapat Road
Pune 411016
Tel: +91 (20) 6703 0000
Fax: +91 (20) 6703 0008

USA
Persistent Systems, Inc.
2055 Laurelwood Road, Suite 210
Santa Clara, CA 95054
Tel: +1 (408) 216 7010
Fax: +1 (408) 451 9177
Email: info@persistent.com

Persistent

www.persistent.com