Behörde / Institution
Hochschulrechenzentrum der Philipps-Universität Marburg
Projektname
argunix
Verwaltungsebene
Bildungseinrichtung
Website / URL
https://www.uni-marburg.de/de/hrz
Einreichungskategorie
Fachverfahren
Möchten Sie das Projekt noch in einer zweiten Kategorie einreichen?
Interne Verwaltungsanwendungen
Projektbeschreibung
argunix ist eine Open-Source-Software zur Erstellung gehärteter, reproduzierbarer Container-Images mit Nix. Als Nix-native CI-Pipeline unterstützt sie GitLab, GitHub sowie Forgejo/Gitea und entkoppelt den Build-Prozess von einer bestimmten Code-Hosting-Plattform. So können Verwaltungen und Unternehmen ihre Entwicklungsinfrastruktur flexibel betreiben, ohne bei einem Plattformwechsel ihre Build-Pipelines neu aufsetzen zu müssen. Der besondere Ansatz von argunix besteht darin, dass sowohl die CI-Pipeline selbst als auch die erzeugten Container vollständig deklarativ und reproduzierbar sind. Zusätzliche Konfigurationswerkzeuge entfallen, wodurch Betrieb und Wartung vereinfacht werden. Gleichzeitig enthalten die mit Nix erzeugten Images nur die tatsächlich benötigten Komponenten. Das reduziert Angriffsflächen, minimiert den Aufwand für Sicherheitsprüfungen und stärkt die digitale Souveränität. Die Build-Logik wird zentral verwaltet und kann plattformübergreifend genutzt werden. Bauen und Testen erfolgen innerhalb derselben Pipeline, wodurch konsistente und nachvollziehbare Ergebnisse entstehen. Die gesamte Konfiguration ist versionierbar, auditierbar und sowohl in Cloud- als auch in On-Premises-Umgebungen einsetzbar. Ein praktischer Einsatz erfolgt derzeit an der Universität Marburg, die mit argunix gehärtete Container-Images für verbreitete Anwendungen wie PHP und MariaDB evaluiert. Automatisierte Integrationstests stellen sicher, dass die erzeugten Images vor der Bereitstellung korrekt funktionieren. Da argunix vollständig auf offenen Standards und Nix basiert, entstehen keine zusätzlichen Herstellerabhängigkeiten. Die Lösung ermöglicht eine sichere, wartbare und plattformunabhängige Build-Infrastruktur und zeigt, wie Open Source die Entwicklung und Bereitstellung moderner Software langfristig souverän und effizient gestalten kann. Eine öffentlich einsehbare argunix-Instanz läuft unter https://argunix.nix-consulting.net/.
Beschreiben Sie den technischen Innovationsgrad des Projektes und den Beitrag zur Verwaltungsmodernisierung.
Die Innovation von argunix liegt nicht in Nix selbst, sondern darin, dessen Deklarativität und Reproduzierbarkeit auf zwei Ebenen nutzbar zu machen: für die CI-Pipeline und für die damit gebauten Container-Images. Daraus folgen mehrere technische Effekte: Die CI verliert ihren Orakel-Charakter. Weil Nix-Builds reproduzierbar sind, lassen sich Pipeline-Fehler lokal nachstellen – die Pipeline bestätigt nur, was lokal ohnehin prüfbar ist. Schnelle Zyklen, dauerhaft release-fähiger Stand. Images, an denen sich wenig geändert hat, werden teils in Minuten neu gebaut, getestet und in die Registry geladen. Der Hauptzweig bleibt so dauerhaft release-fähig – wichtig für schnelle Reaktion auf Sicherheitslücken. Messbar kleinere, sicherere Images.Beispielprojekte auf openCode (https://gitlab.opencode.de/oci-community/images/applicative-systems/images) belegen das: das Kubernetes-Werkzeug-Image schrumpft von 296 auf 61 MB, prometheus von 161 auf 59 MB, memcached von 32 auf 13 MB. SBOM ohne nachträgliches Scannen. Die Nix-Bauanleitung eines Images führt den vollständigen Abhängigkeitsbaum explizit mit. Die SBOM ist damit keine Schätzung

eines Scanners, sondern exakte Ableitung aus dem Bauplan – redis: 8 statt 40 Komponenten, memcached: 5 statt 45.
Weniger Pflegeaufwand pro Image. Eine Image-Definition umfasst wenige Dutzend Zeilen lesbarer Konfiguration statt hunderter Zeilen generierten Shell-Codes – memcached 37 statt 442 Zeilen, redis 75 statt 467.
Der Beitrag zur Verwaltungsmodernisierung: Weil eine einzige Pipeline GitLab, GitHub und Forgejo gleichzeitig bedient, wird der schrittweise Umstieg auf selbst betriebene, europäische Plattformen zum beherrschbaren Prozess statt zum riskanten Großprojekt. argunix ist auf Interoperabilität ausgelegt: gleichwertige Unterstützung der CI-Konzepte und APIs aller drei Plattformfamilien, Auslieferung über OCI-Registries und signierte Caches als zweiter, herstellerunabhängiger Kanal.

Welchen ökomischen Nutzen hat das Projekt?

Kein zusätzliches Config-Tooling. Weil die Pipeline selbst deklarativ ist, entfallen Ansible/Puppet & Co. für ihren eigenen Betrieb – das spart Schulungs- und Wartungsaufwand für ein ganzes Werkzeugregal.
Keine Lizenzkosten für argunix. GPL-3.0-or-later, beliebig viele Instanzen, gebührenfrei – relevant besonders für Kommunen mit knappen Budgets.
Vermiedene Anbieterabhängigkeit. Weil die Code-Hosting-Plattform frei austauschbar ist, bleibt die Verhandlungsposition gegenüber Anbietern erhalten; ein Plattformwechsel erfordert keinen Neuaufbau der Pipeline.
Wegfall von Doppel-Infrastruktur. Eine Pipeline bedient mehrere Code-Hosting-Plattformen und ist zugleich die Testinfrastruktur – das spart Hardware, Wartung und Personal.
Weniger CI-Pflege für Image-Maintainer. Durch die Integration der CI-Konzepte verbreiteter VCS können sich Maintainer auf Bauanleitungen, Integrationstests und Aktualisierung ihrer Images konzentrieren, statt komplexe CI-Konfigurationen zu pflegen.
Geringere Laufzeit- und Speicherkosten. Die genannten Größenreduktionen (z. B. 296→61 MB) übersetzen sich direkt in geringere Registry-, Transfer- und Laufzeitkosten.
Sicherheit und Compliance ohne Zukauf. Reproduzierbare Builds und eingebaute SBOM ersetzen kommerzielle Supply-Chain-Sicherheitsprodukte; die geringe Komponentenzahl pro Image senkt zusätzlich den Aufwand für nachgelagertes Compliance-Tooling.
Eingesparte Rechenzeit. Feingranulares Caching baut nur das neu, was sich geändert hat, und teilt Ergebnisse über signierte Caches – das senkt Energie- und Infrastrukturkosten.
Echter Support-Wettbewerb. Weil alles offen und wartbar ist, kann jeder qualifizierte Dienstleister Support leisten – kein Anbieter-Monopol. Die schlanke Architektur (ein Dienst mit eingebetteter Datenbank) senkt Betriebsaufwand und Ausfallzeiten zusätzlich.

Beschreiben Sie die Nachhaltigkeit der Lösung.

Langfristige Reproduzierbarkeit. Jeder Input ist per Hash gepinnt; derselbe Commit erzeugt heute wie in Jahren ein bit-identisches Image – entscheidend für Verwaltungssysteme mit Aufbewahrungs- und Nachweispflichten.
Wartbarkeit unabhängig von Einzelpersonen. Die textbasierte, versionierte Pipeline-Konfiguration dokumentiert sich selbst und mindert Wissensverlust bei Personalwechsel.
Kein Anbieter-Risiko. Der Quellcode steht unter GPL dauerhaft offen auf der europäischen, gemeinnützigen Plattform Codeberg. Fielen die ursprünglichen Maintainer aus, könnten Dritte die Pflege übernehmen – anders als bei proprietärer Software.
Geprüfte Auslieferung. Vor der Veröffentlichung wird jedes Image automatisiert gestartet und auf korrekte Funktion geprüft (im Beispiel des Learning-Management-Systems ILIAS bis zum Abruf der Login-Seite). Das verhindert, dass fehlerhafte Stände produktiv gehen.
Rückfluss in die Gemeinschaft. Verbesserungen können als Upstream-Beiträge zurückfließen. Die geplante Integration mit der europäischen Sicherheitsplattform DevGuard (von openCode bereits eingesetzt) und die offenen Beispielprojekte auf openCode stärken ein gemeinsames Ökosystem.
Weg zu container.gov.de. Nach Abschluss der Evaluation durch die Universität Marburg

können die mit argunix erzeugten, regelmäßig aktualisierten Images öffentlich über container.gov.de bereitgestellt werden. Weitere Anwendungen aus Verwaltung, Bildung und Forschung könnten diesem Modell folgen.
Ressourcenschonung. Feingranulares, geteiltes Caching, schlankere Images und der Abbruch überholter Berechnungen senken den Energieverbrauch; die schlanke Architektur kommt mit deutlich weniger Hardware aus.
Tragfähige Trägerschaft. Die gemeinsame Pflege durch Applicative Systems GmbH und Universität Marburg verbindet wirtschaftliche Stabilität mit wissenschaftlicher Fundierung und akademischem Nachwuchs.

Wie trägt das Projekt zur Stärkung der Digitalen Souveränität bei?

Die zentrale Souveränitäts-Idee von argunix: Weil argunix eine Nix-native CI-Pipeline für Container-Images ist, gilt dieselbe Deklarativität und Reproduzierbarkeit sowohl für die Pipeline als auch für die von ihr gebauten Artefakte – eine Bau-Entscheidung zahlt zweimal auf Souveränität ein.
Die Pipeline braucht kein zusätzliches Fremd-Tooling. Keine separaten IaC- oder Konfigurationswerkzeuge für den eigenen Betrieb bedeuten weniger Software, der man vertrauen und die man absichern muss – weniger Abhängigkeiten, mehr Kontrolle im eigenen Betrieb.
Die Images enthalten nur, was gebraucht wird. Kein unnötiges Linux-Userland, keine Shell – kleinere Angriffsfläche und eine SBOM, die direkt aus dem Bauplan folgt statt nachträglich erscannt zu werden.
Konkret setzt das drei weitere Hebel der digitalen Souveränität in Bewegung:
Freie Austauschbarkeit der Code-Hosting-Plattform. argunix unterstützt GitLab (Cloud und selbst gehostet, etwa openCode/ZenDiS), GitHub, GitHub Enterprise und die Forgejo-Familie gleichwertig. Eine Behörde ist nicht an einen einzelnen, oft außereuropäischen Anbieter gebunden und kann wechseln, ohne die Pipeline neu aufzubauen – ein erklärtes Ziel von ZenDiS und Bundesregierung.
Vollständige Betreibbarkeit in eigener Verantwortung. argunix ist quelloffen (GPL): keine verborgene Logik, keine erzwungene Cloud-Anbindung, keine Telemetrie. Der Betrieb ist auch in abgeschotteten Netzen möglich – Bau-Rechner brauchen nicht einmal offene eingehende Ports.
Selbst gehostete, vertrauenswürdige Auslieferung. Eigene Caches lassen sich trivial auf etablierter Infrastruktur (z. B. S3) selbst hosten, sind stark zugriffsbeschränkt und signierbar – ein Kanal neben der Container-Registry.
Grundlage sind offene Werkzeuge: Nix, Codeberg als europäische Plattform und die GPL-Lizenz. So wird Souveränität zur technischen Eigenschaft statt zum Versprechen.

Projektwebsite

https://codeberg.org/tfc/argunix

Umsetzungspartner

Applicative Systems GmbH (https://applicative.systems/de), Universität Marburg (https://www.uni-marburg.de/de)

Zusätzliche Dokumente

ilias_images.pdf
opencode_images.pdf