A DevOps Approach for Cyber-Physical System-of-Systems Engineering through Arrowhead

Csaba Hegedűs, Pál Varga
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics
2 Magyar Tudósok krt., Budapest, Hungary, H-1117
Email: hegeduscs@gmail.com, pvarga@tmit.bme.hu

Attila Frankó
AITIA International Inc., Industrial IoT Division
48-50 Czetz János str., Budapest, Hungary, H-1039
Email: afranko@aitia.ai

***Abstract*—The stakeholders in the Cyber-Physical System-of-Systems (CPS, SoS, CPSoS) domains need to adopt current methodologies that enable reliable but also flexible and timely completion of development, integration, deployment, operation, and even maintenance-related tasks. The so-called DevOps (Development & Operations) approach has been proven in various other domains of IT-related service completion and operation. Both the technology set and the management approaches of CPS and SoS practitioners can benefit the adoption of the DevOps toolchain success stories – the everyday life of software, cloud and IT-service development, and operations-focused companies. This paper proposes an expansion of the application area of the DevOps models, toolchains, and even utilities to the CPS and SoS domains. It proposes to create an abstraction of CPS devices as reusable infrastructure resources, which can be then automatically re-purposed for new workflow tasks, as well as to fill the remaining DevOps tooling gaps with actual Systems of the Arrowhead Framework.**

I. Introduction and Motivation

CLOUD and related automation technologies have changed how software development and infrastructure operations function in enterprises. Agile DevOps (Development & Operations) [1] and security-enhanced DevSecOps [2] methodologies enable flexible, reliable and timely completion of IT-related processes. This is realized by "Dev" (-plan-code-build-test-release-) and "Ops" cycles (-release-deploy-operate-monitor-) circling into each other. The related supporting software stacks provide means of working for companies for developing and operating their IT services and infrastructure [3].

There is now a plethora of solutions available (let that be open source or off-the-shelf) that can be combined in numerous ways to create the technology stack required for the agile DevOps practice. The underlying IT infrastructure hosting these enterprise services is also mostly assembled in a *hybrid cloud manner*, consisting of (i) legacy, (ii) on premises and private or (iii) public cloud resources.

Still, such hardware heterogeneity and software diversity are handled fairly well within these modern Dev(Sec)Ops tool chains. Here, among others, *continuous configuration automation* tools play a significant role in harmonizing and creating the Infrastructure as Code (IaC) abstraction of hybrid cloud infrastructures. Furthermore, *virtualization and containerization* technologies enable the creation of container images, that are easy to maintain and deploy. Finally, *deployment automation* tools make sure that the envisioned software architecture runs "smoothly" using the developed, containerized applications.

978-3-903176-32-4

Meanwhile, industrial and manufacturing automation related IoT systems also face similar, but unresolved challenges. These are brought on by the diversity of hardware resources and software development practices even more so that it is experienced within the IT services domain. IT and OT (Operational Technologies) are converging in their functionalities to form Industrial IoT through realizing Cyber-Physical System-of-Systems (CPSoS). The motivation of this paper is to show how IT and OT development and operations practices can converge based on the DevOps approach.

There are some clear challenges that appear when mapping DevOps methodologies and toolsets to *embedded* and *ambient* systems. This is due to the specific nature of these projects, such as the source code is tied to a specific hardware, which has limited resources, or the piece of hardware may also get produced in low quantities. Therefore automating the tests is often painful and lacks standard tools that could ease this process, while testing itself is considered the most important part of an embedded project. In many cases – where it is available – simulators are used to test the software, however only bigger market players provide such solutions. Additionally, waterfall-like models are often applied instead of agile ones, since embedded firmware usually runs for years without any modifications. This is especially true for safety-critical systems, where the current software version has to be certified, leaving no room for frequent changes regarding requirements. Moreover, low-powered IoT nodes (e.g. sensors) usually aren't IP capable devices (yet), and therefore we cannot usually assume that they're accessible remotely via any network. While this might still be possible for industrial nodes (within e.g. factory sites), it keeps to be challenging in rural environment, where permanent power supply is not available.

On the other hand, some elements of the DevOps processes are already integral parts of modern embedded projects, i.e. code quality checks. This is because the source code often has to meet the requirements of a coding standard in industrial projects (mission-critical included). Generally, adopting further DevOps approaches and tools for embedded projects could turn them more efficient in many ways, due to the high level of automation. Although it is a widely accepted approach, the lack of appropriate commercial tool chains makes it hard to create a global standard process.

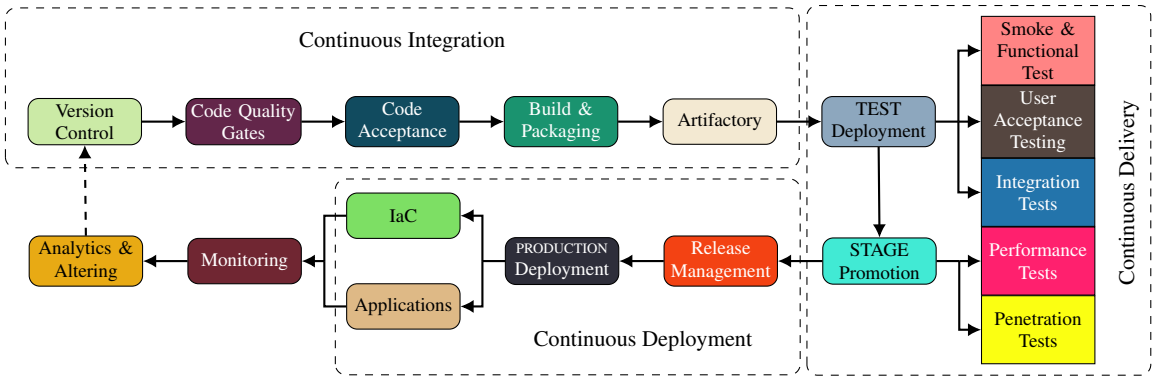

Fig. 1. A typical DevOps process for Cloud-based systems – including various sorts of continuous testing. The process circles around.

The contributions of this paper are the following:

- It proposes to extend DevSecOps methodologies, practices and tools to the CPSoS domain.
- It proposes a concept, where a configuration management middleware layer (i.e. the Arrowhead framework), together with basic DevOps practices and tool chains create an abstraction of the industrial IoT and CPS devices as reusable infrastructure resources, which can then be automatically re-purposed while provisioning new workflows.
- Based on this concept, this paper also suggests augmentations for the Arrowhead framework with (i) continuous configuration automation and (ii) deployment automation capabilities and (iii) interfaces towards other engineering tools for creating an end-to-end DevOps tool chain in the industrial IoT world.

II. Related Works

A. Toolchains for Cyber-Physical Systems

There are several development and operational best-practices available for the manufacturing industry [4] – which is one of the main endorsers of CPSoS approaches. The IEC 81346 standard [5] defines the engineering process for automation systems, which have been extended by the Arrowhead Tools initiative. Extending the standard, this initiative proposes various toolchains for multi-stakeholder System-of-Systems development and integration [6] [7]. The main motivations of that work include the idea of flexible tool-usage in the diverse ecosystems of industrial IoT, and that input and output data of the chosen tools can be used as automatically (without human interaction) as possible. The Arrowhead Tools project has already validated this model through use-case examples [8].

On the other hand, despite its widely acclaimed advantages (i.e., agility, efficiency, time-to-market, etc.) DevSecOps-based methodology and tooling has seemingly been neglected by the CPSoS domain. So far there are merely a few proposals towards this direction. One on the pioneers is the "Industrial DevOps" initiation [9], which briefly describes a continuous adaption and improvement process, describing the elements and flows through organizational rather than technical processes. Querejeta et. al. propose their approach towards using DevOps principles when working with Digital Twins in Cyber-Physical Production Systems in [10]. This should be only the beginning, since DevOps provides a myriad of solutions for making the engineering process flowing better – even for production systems, or for the wider CPSoS domain.

According to a current comprehensive survey [3] the major DevOps tools can be categorized the following way based on their main functionalities: knowledge sharing; source code management; build process; continuous integration; deployment automation; monitoring and logging. There are various other tool taxonomies that exist [11], [12], [13], where the stages are defined based on the different cultures and philosophies of the teams created them.

This paper considers a relevant DevOps end-to-end pipeline as Figure 1 depicts. Here, we assume a generic tool chain that thrives for automating most of the development, integration, testing, deployment and release management tasks. These toolchains are very easy to assemble on a high level – with a mixture of open source and COTS (Commercial Off-The-shelf) products –, but in reality it is closer to art to actually implement it well. Nevertheless, in Industry 4.0, we need to target such a solution for industrial and automation IoT as well.

B. DevOps Principles and Tools

In a well-designed DevOps pipeline, Continuous Integration (CI) tools and processes enable developers to collaborate and produce good quality code, in an agile manner. This is aided by version control tools and methods to reach consensus on the code base (i.e. merge requests and collective code reviews). Popular tools are Gitlab [14], Microsoft DevOps [13] or Github Enterprise [15].

Moreover, automated code quality gates are usually hooked up within the CI pipeline, enabling the developers to fix the most common mistakes, easily identifiable by static code analysis. These checks range from primitive coding issues to conferring its dependencies with known vulnerabilities databases. Popular products for code quality checks include e.g. SonarQube [16], while JFrog Xray [17] is a good example for containerized applications.

After code acceptance, the built binaries, images or containerized applications (built usually by a build server, e.g. Jenkins [18] or Bamboo [19]), deployment automation tools pick up the developed software artifacts from the artifactory

storage (e.g. JFrog [17]). Here, we envision a standard environments' setup, with formalized code promotion process through lower environments to production. Various automated test pipelines can be created and executed related to, without being exhaustive, (i) functional / component level, (ii) integration tests, (iii) user acceptance, (iv) performance and (v) penetration testing schemas.

Finally, release management needs to take care of managing the production environment(s). Here, it is essential to see the separation between infrastructure code and application code in modern DevOps practices. In modern ecosystems, where *Infrastructure as Code* (IaC) tools are deployed (see II-B2), all infrastructure related processes are automated and designed via descriptive programming, and applications are deployed automatically as well, on said infrastructure.

1) Virtualization and Containerization: This paper considers virtualization and containerization as the key driver of modern processing and cloud technologies. These technologies enable resources to be shared between workloads and also make the deployment of new software and configuration easier. A containerized (e.g. web) application can be deployed automatically, in any number of instances, with customized configuration, connectivity, persistence, with full required run time also properly set up.

Modern virtualization techniques rely on hardware level isolation, managed mostly by the processors themselves, run on specialized hypervisor OS. Meanwhile, containerization relies mostly on operating system (OS) level separation, and containers mostly share the same OS kernel. It is worth noting that these two solutions require very different capabilities from applications and their hosts [20].

In embedded programming, such advanced application packaging and execution silo mechanisms are naturally not available (yet), neither in classic production control systems. However, embedded firmware images and e.g. soft PLC (Programmable Logic Controllers) software can usually be built for individual devices in an automated way, and then flashed or configured remotely. IP networking access for industrial IoT devices is one of the main underlying assumption of the Industry4.0 movement as well. This paper supposes therefore that industrial and automation IoT devices and control units can be fully configured, updated and managed remotely, via IP network.

2) Infrastructure as Code (IaC): Modern DevOps practices mostly revolve around being able to define whole underlying infrastructures (even in hybrid cloud scenarios) as high level script code, which is then automatically compared with currently existing setup and upgraded to the desired state through planning and approval. Here, various drivers and plugins have been developed within these IaC tool chains to support bare metal, on-prem cloud and public cloud IaaS, PaaS and SaaS infrastructure components. These IaC tools take care of the following tasks:

- Primitive resource provisioning (availability zones, resource and scaling groups, virtual machines, etc.)
- PaaS and SaaS resource provisioning (managed databases, various gateways & proxies, etc.)
- Networking configuration (virtual subnets, IP addresses, firewalls, load balancing, etc.)

IaC, CCA (Continous Configuration Automation) and DA (Deployment Automation) tools are often mixed together, or categorized differently than how it is presented here. However, for our purposes, this categorization helps us better visualize and present where the proposed Arrowhead solution lies.

Terraform [21] is a popular IaC tool that allows DevOps engineers to describe the target infrastructure using YAML descriptors. Terraform assembles a resource graph, and confers target architecture with current state, and plans the difference. This plan can then be approved and executed on large scale. Other popular alternatives are Chef [22] or Salt [23].

3) Continous Configuration Automation (CCA): Continous Configuration Automation (CCA) tools enable the remote configuration of various hosts (e.g. bare metal or virtual machines, etc.), and in most cases they are used together within IaC plans. These tools configure hosts via remote access, most prevalently over Secure Shell (SSH) for Linux hosts.

As for containerized environments, CCA tools are often used to execute regular workflows over e.g. the Kubernetes [24] or Openshift [25] APIs, i.e. automating operations tasks (e.g. scaling or upgrades).

Ansible [26], which is a highly popular CCA tool, defines "playbooks" that are YAML files expressing configurations, deployment, and orchestration, and allow Ansible to perform operations on managed nodes.

4) Deployment Automation (DA): Deployment automation allows to move your software to lower (i.e. development or test) and production environments by using automated processes. This leads to repeatable and reliable deployment processes across the software delivery cycle.

In a generic DA tool chain, developers create containerized applications, and place it in a build artifactory using build servers, and tools. After that, the DA pipeline can be invoked, and deployment starts with the available artifactory image. Complex DA systems, such as the Openshift [25] ecosystem (with Kubernetes [24] inside), automate container deployment in its full extent, i.e. every step after a new version of the image is pushed to the artifactory. A well-built CICD platform (with DA at its end) enables quality gates, automated test pipelines and release approvals, and takes care of the code promotion process as well. Usually, it is can also be in scope for DA systems (e.g. Kubernetes "extended" with Istio [27]), to facilitate the (micro)service mesh architecture pattern.

One interesting and highly relevant concept within DA solutions (such as in Kubernetes), is node tainting and node affinity [28]. In general, these mechanisms allow the cluster management system to decide where a given workload (or *pod*) can be deployed. DevOps engineers can label nodes with various details, and can also specify requirements for workloads on which nodes the workload can actually be deployed. This concept can be used to dedicate certain nodes for a workload, or mark nodes that have special hardware capabilities and only those can run certain workloads. In a full Industry 4.0 corporation, such handling such device heterogeneity within its platform is essential.

III. A DevOps Approach for System-of-Systems Engineering

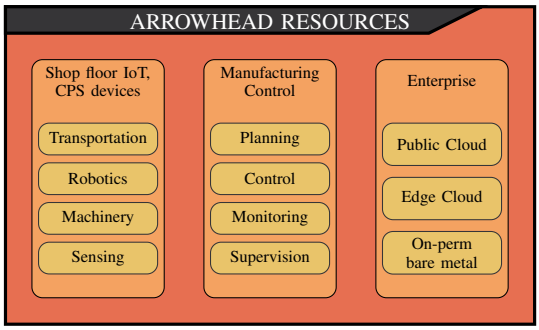

Fig. 2. Various types of Arrowhead resources in an enterprise

This paper introduces CCA and IaC to industrial and automation IoT deployments. Our solution is targeting a world, where an entire shop floor can be reconfigured instantaneously, through automated production code promotion to the CPSoS. The presented concepts and tool chains aim to alleviate hardships brought on by the manual, and individual device update configuration tasks that are so prevalent in today's manufacturing world. The main assumptions we rely on:

1) Devices should be handled as reconfigurable, reusable resources, having various levels of capability.
2) These devices are all Internet Protocol (IP) capable, and their controlling software and system configurations can be modified remotely.
3) Manufacturing automation could be achieved with individual devices having appropriate software version and configuration setup (the required physical layout changes are considered as pre-deployment tasks).
4) Developers who work on individual CPS and IoT device software can code and test on lower environments, production grade software can be promoted via release management to the manufacturing floors.
5) Industrial and automation IoT architectures can be service-oriented, and all network communications can be realized with IP based communications (even hard real time applications).
6) Runtime governance of the CPSoS and manufacturing workflow management can be achieved, for example via the Arrowhead framework and its related engineering tool chains [6], [29], [30], [31].

Furthermore, this paper also proposes the creation of a unified infrastructure abstraction layer that handles *all resources* in an enterprise, not just shop floors or IoT devices. The scope for this paper is depicted in Fig. 2. In an Industry 4.0, all development and operations tasks should be unified, regardless whether it is the shop floor itself, the Manufacturing Execution System layer (MES) or enterprise resources are involved.

The proposed tool chain enables developers to work on their respective development and continuous integration (CI) tools, but the assembled executables (binaries, system images, container images, etc.) would all still "land" in the corporate artifactory (single source of truth). Here, annotations and taints put on the software and configuration deliverables are essential. Not just versioning, or file naming, but other details needs to

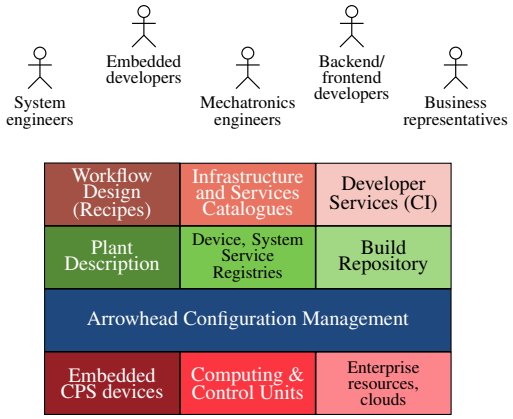

Fig. 3. Arrowhead Configuration Management

be affiliated with artifacts, so that a deployment middleware can recognize their potential purpose and future usage.

Based on these taints node tolerations, the Arrowhead configuration management middleware will take care of setting up the required infrastructure resources in order to run the assembled workflows, with the most up-to-date and appropriate device software and configuration. Fig. 3 depicts the proposed tool chaining. The purpose of this solution is to abstract enterprise and automation resources together and homogeneously as a Platform as a Service (PaaS) to developers. Developers should be able to utilize the available resources of the enterprise on the lower, or even on the production environments via *self-service*. They should be able to push and test their code to embedded systems, industrial control units or enterprise resources via the same platform. Moreover, but as equally important, system architects should also be able to choreograph resources via the same platform.

A. Resource Categories in Industry 4.0 Applications

This paper proposes a categorization for devices that are present in industrial and automation IoT and enterprise solutions, based on their capability. Table I depicts *an example* for this classification, with the most important aspects considered. Further considerations need to be on CPS and IoT device categorization, especially in the Arrowhead context [32], [33], but this is not in scope for this paper, as our main focus here is related to remote software and configuration management.

1) *Restrained embedded device*: devices that have very limited resources and network connectivity. These devices are usually sensors or small actuators, and they serve a single purpose. Remote updates to the firmware should still be possible, though.
2) *Basic embedded device*: devices the are used for embedded programming, but do not suffer from resource constraint. They still execute simple tasks, but the device capabilities enable for security [34], software silo mechanisms and remote updates.
3) *Enhanced embedded device*: embedded devices that have fully fledged OS kernels, and can execute multitasking or even simple containerization.
4) *Computing control units (CCU)*: traditional controlling layer in manufacturing architectures, with fully

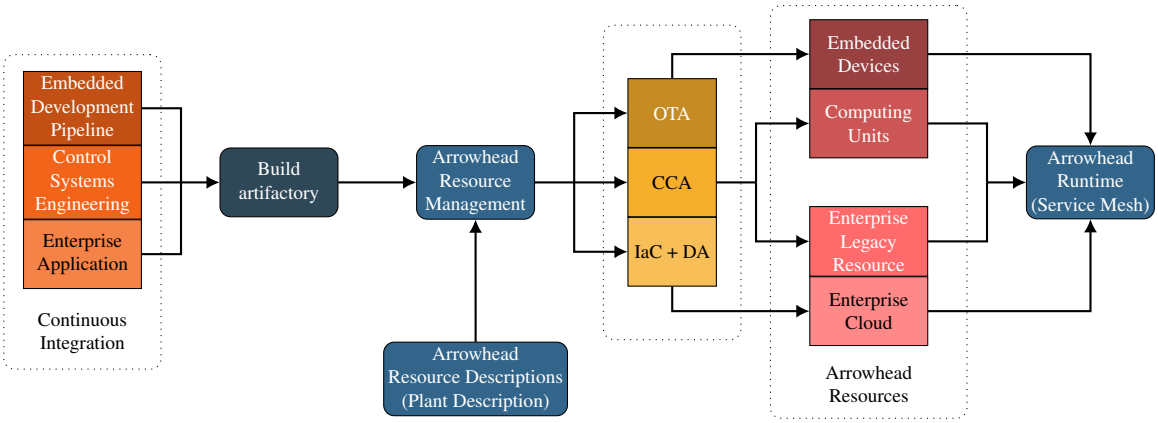

Fig. 4. Proposed DevOps pipeline for Industry 4.0 – Note: OTA: "Over-the-Air" - CCA: "Continous Configuration Automation" - IaC: "Infrastructure as Code" - DA: "Development Automation"

fledged operating systems. Cryptography services may be implemented as well. Real time IP-based networking is needed between CCU and embedded systems on the shop floors.

5) *Enterprise resources*: legacy enterprise applications, usually hosted on premises, maybe virtualized.
6) *Enterprise cloud*: private or public clouds, hosting containerized app's on virtualized infrastructure.

All these different types of resources need to be managed and configured by a central resource management middleware, so that changes can be deployed automatically, if selected. This can be achieved by annotating both the software artifacts and the devices (resources) with details related to (i) (supported) device types, (ii) (required) target device functionality, (iii) (target) device location - both physical and logical and (iv) associated workflow where this artifact will be required.

For the last two device types, CCA and DA tools are already there with industrial best practices crystallized out. However, for embedded systems or control units, the basic necessities for the DevOps and IaC practices are currently not available in public tool chains. There is (usually) no containerization involved, neither are standard mechanisms available for configuring fleets of industrial IoT devices.

This paper proposes standardizing the mechanisms of embedded systems to update their firmware and configuration remotely, based on the assumptions made in section II-B1. Such a requirement is feasible, and can create the same homogeneous remote connectivity platform as SSH for Linux (or Windows RDP) is considered for cloud or on-premises enterprise resources. Following the industrial practice, we commonly refer these methods and techniques with the historical term "Over The Air" (OTA), even if these are wired connections. Open source OTA tools and IoT fleet management tools are, however, already available, even in open source format. For example, Eclipse Hawkbit [35] or Mender.io [36] are popular solutions of software version and configuration management via OTA.

B. DevOps Toolchain for Industry 4.0

Figure 4 summarizes the proposed DevOps toolchain. Here, the CI pipelines (embedded development, industrial control & systems engineering, enterprise development) can all be customized, to fit the domain needs. Basic testing and collaboration tools can be incorporated together with the appropriate build tools. Moreover, this solution supposes one enterprise grade artifactory, and a related artifact annotation process.

From here, a complex DevOps toolchain is defined that comprises of various tools, depending on the device (resource) types where the artifact can be deployed. As shown, these refer to OTA, CCA and IaC with DA tool chains. However, as these tools require configuration on their own, an additional middleware layer is required to mask the infrastructure heterogeneity.

Using this Arrowhead middleware, the DevOps workflow for a developer would be independent from what is being developed. The person must be able to annotate the artifact, for example that it is a (i) a specific restrained embedded system type firmware, (ii) targeting pressure sensors (iii) on press bench workstations 4-10, (iv) as part of workflow 3 ("sedan door assembly"). Later on, for this artifact, as it is an embedded firmware one, OTA will be used by the Arrowhead configuration middleware to batch update the firmware on the target nodes *(i.e. update firmware and configuration on all pressure sensors on press bench workstations 4-10 before sedan door assembling workflow begins)*. Actual workflows can then be choreographed and executed then on using other Arrowhead tool chain solutions [6], [31], [37].

IV. Conclusions and Future Work

Agile DevOps practices, technologies, toolchains and frameworks enabled new form of software development and deployment. Although, this approach is already utilized by enterprises, it has not been adopted for the lower levels of industrial CPSoS, due to certain reasons such as: heterogeneity of devices and related software. This paper described a concept of combining standard DevOps tools and practices with the Arrowhead framework to enable a new workflow for development and deployment of industrial IoT and CPS devices. Next steps in this work includes creating PoC demonstrations to showcase an Industry 4.0 enterprise and how it can benefit from the concept drawn here.

TABLE I. MAIN TYPES OF CYBER-PHYSICAL SYSTEM ELEMENTS (COLUMNS) – AND SOME OF THEIR DISTINGUISHING FEATURES (ROWS)

	Restrained embedded	Basic embedded	Enhanced embedded	Basic Computing and control unit	Enterprise resources	Enterprise cloud
IP Connectivity	Single Interface TCP/IP	TLS v1.2	TLS v1.2	Multiple Interface	Multiple Interface	Virtualized Networking
Remote update and configuration	OTA	OTA	SSH + SCP	SSH + SCP	SSH + SCP	containerized workload mgmt
Operating system	—	—	Real-time embedded OS	UNIX-like, Windows, others	UNIX-like, Windows	Hypervisor
Applications	single task	single taks	multitasking	run by OS	run by OS	run by container management
Security	none	firmware encryption	firmware encryption, opt. TPM	OS security SW measures	HW, OS, app. level security	Central secret management

ACKNOWLEDGMENT

This research was funded by the European Commission and the Hungarian Authorities (NKFIH) through the *Arrowhead Tools* project (EU grant agreement No. 826452, – NKFIH grant 2019-2.1.3-NEMZ_ECSEL-2019-00003.

REFERENCES

[1] B. B. N. de França, H. Jeronimo, and G. H. Travassos, "Characterizing devops by hearing multiple voices," in *Proceedings of the 30th Brazilian symposium on software engineering*, 2016, pp. 53–62.

[2] P. Abrahamsson, G. Botterweck, H. Ghanbari, M. G. Jaatun, P. Kettunen, T. J. Mikkonen, A. Mjeda, J. Münch, A. N. Duc, B. Russo *et al.*, "Towards a secure devops approach for cyber-physical systems: An industrial perspective," *International Journal of Systems and Software Security and Protection (IJSSSP)*, vol. 11, no. 2, pp. 38–57, 2020.

[3] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of devops concepts and challenges," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–35, 2019.

[4] D. Kozma and P. Varga, "Supporting digital supply chains by iot frameworks: Collaboration, control, combination," *Infocommunications Journal*, vol. 12, no. 4, pp. 22–32, 2020.

[5] ISO-IEC, *Industrial systems, installations and equipment and industrial products — Structuring principles and reference designations — Part 1: Basic rules*, Standard 81 346, 2009.

[6] G. Urgese, P. Azzoni, J. v. Deventer, J. Delsing, and E. Macii, "An engineering process model for managing a digitalised life-cycle of products in the industry 4.0," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.

[7] G. Kulcsár, M. S. Tatara, and F. Montori, "Toolchain modeling: Comprehensive engineering plans for industry 4.0," in *The 46th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2020.

[8] R. Venanzi, F. Montori, P. Bellavista, and L. Foschini, "Industry 4.0 solutions for interoperability: a use case about tools and tool chains in the arrowhead tools project," in *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2020, pp. 429–433.

[9] W. Hasselbring, S. Henning, B. Latte, A. Möbius, T. Richter, S. Schalk, and M. Wojcieszak, "Industrial devops," in *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2019, pp. 123–126.

[10] M. U. Querejeta, L. Etxeberria, and G. Sagardui, "Towards a devops approach in cyber physical production systems using digital twins," in *International Conference on Computer Safety, Reliability, and Security*. Springer, 2020, pp. 205–216.

[11] AWS, "Practicing continuous integration and continuous delivery on aws," https://docs.aws.amazon.com/whitepapers/latest/practicing-continuous-integration-continuous-delivery.

[12] Red Hat, "A primer on devops pipeline: Continuous integration continuous delivery (ci/cd)," https://www.redhat.com/architect/primer-devops.

[13] Microsoft, "Devtest and devops for microservice solutions," https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/dev-test-microservice.

[14] Gitlab, "About us," https://about.gitlab.com/, 2021.

[15] Github, "Enterprise," https://github.com/enterprise, 2021.

[16] SonarQube, "Code quality and code security," https://www.sonarqube.org/, 2021.

[17] JFrog Inc, "Product landscape," https://jfrog.com/#products, 2021.

[18] Jenkins, "Build great things at any scale," https://www.jenkins.io/, 2021.

[19] Atlassian, "Bamboo CI," https://www.atlassian.com/hu/software/bamboo, 2021.

[20] J. Turnbull, *The Docker Book*. Turnbull Press, 2014.

[21] Hashicorp, "Terraform IaC," https://www.terraform.io/, 2021.

[22] Chef, "Enterprise Automation Stack," https://www.chef.io/products/chef-infra, 2021.

[23] SaltStack, "Enterprise," https://www.saltstack.com/products/saltstack-enterprise/, 2021.

[24] Kubernetes.io, "What is Kubernetes?" https://kubernetes.io/, 2021.

[25] Redhat Inc., "Openshift," https://www.openshift.com/learn/what-is-openshift, 2021.

[26] Redhat, "Ansible," https://www.ansible.com/, 2021.

[27] Istio, "Service Mesh," https://istio.io/latest/docs/concepts/what-is-istio/, 2021.

[28] Kubernetes, "Node affinity and taining," https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/, 2021.

[29] The Arrowhead consortia, "The arrowhead framework github," https://github.com/arrowhead-f.

[30] P. Varga, D. Kozma, and C. Hegedűs, "Data-driven workflow execution in service oriented iot architectures," in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2018.

[31] D. Kozma, P. Varga, and K. Szabó, "Achieving flexible digital production with the arrowhead workflow choreographer," in *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2020.

[32] A. Bicaku, S. Maksuti, C. Hegedűs, M. Tauber, J. Delsing, and J. Eliasson, "Interacting with the arrowhead local cloud: On-boarding procedure," in *IEEE industrial cyber-physical systems (ICPS)*, 2018.

[33] C. Hegedűs, P. Varga, and S. Tanyi, "A governance model for local and interconnecting arrowhead clouds," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, 2020.

[34] A. Bicaku, C. Schmittner, P. Rottmann, M. Tauber, and J. Delsing, "Security safety and organizationalstandard compliance in cyber physicalsystems," *Infocommunications Journal*, vol. 11, no. 1, pp. 2–9, 2019.

[35] Eclipse, "Hawkbit," https://www.eclipse.org/hawkbit/, 2021.

[36] Mender.io, "IoT OTA SaaS," https://mender.io/, 2021.

[37] O. Carlsson, C. Hegedűs, J. Delsing, and P. Varga, "Organizing iot systems-of-systems from standardized engineering data," in *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2016.