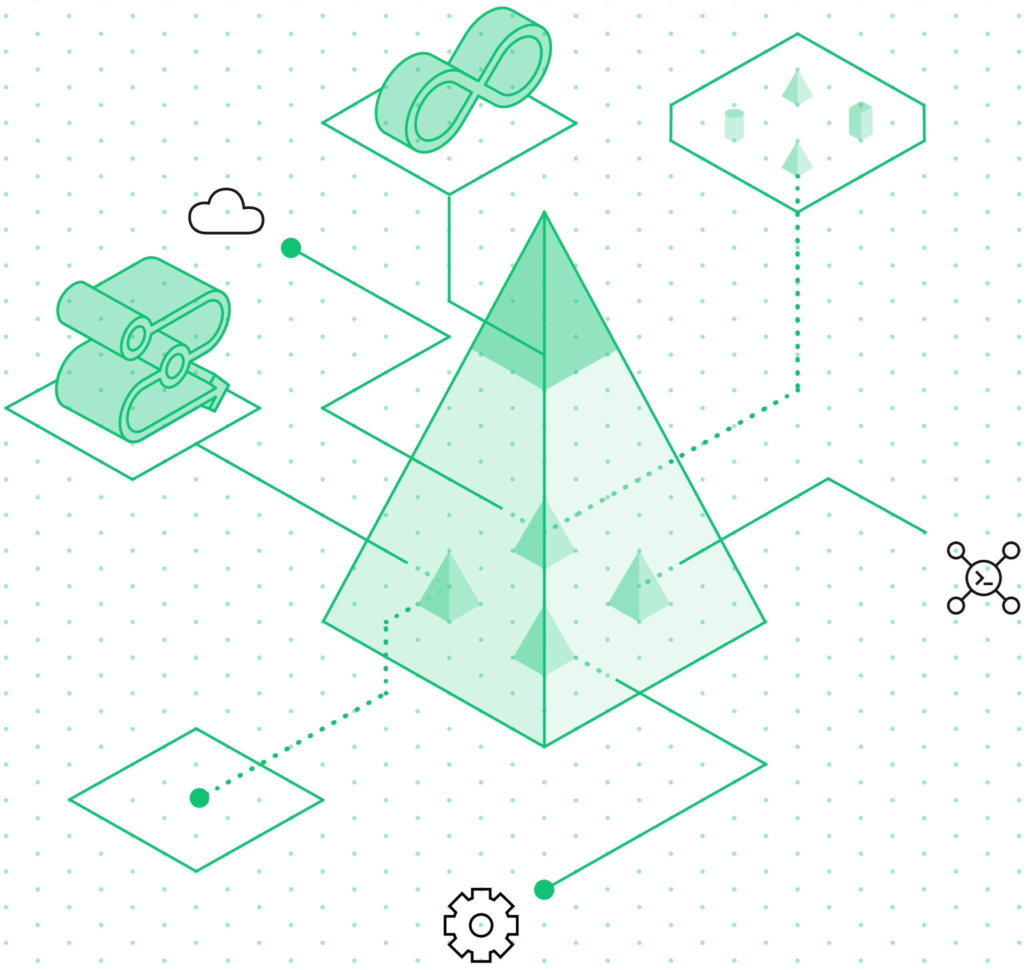
ericsson.com/
5g-core-guide

CI/CD: Continuous software for continuous change

Your guide to CI/CD evolution for telecom software in a cloud-native ecosystem



The cloud-native 5G Core network guide series 2.0

CI/CD today and tomorrow

Continuous integration and continuous deployment (CI/CD) is the cornerstone of true digital transformation. In this guide, we'll share insights into life cycle management (LCM) automation using CI/CD practices.

Contents

Our society is on a journey to digitalization, expanding to almost all industries. Use cases such as Massive and Critical IoT, public-safety solutions and various 5G-enabled services are emerging. This increases the demands placed on your communication network for features and higher performance. Perhaps most importantly, as digitalization affects more and more processes around us, it requires secure, high-quality connections.

With this evolution, innovative communications service providers are asking for speed and agility to launch new features and functionalities in their networks. A continuous flow of new capabilities and improvements – especially with the advent of 5G – is becoming necessary to stay competitive. This is where CI/CD comes in.

In our dialogue with service providers, we have noted different automation ambitions and considerations of how CI/CD will impact their ways of working. We have found that 5G's complexity needs to be handled by the production system, whether it relies on purpose-built hardware, is executed on off-the-shelf hardware in virtualized deployment scenarios or is run in a cloud-native environment. Likely, all of these will coexist.

CI/CD and automated LCM is particularly expected for the cloud-native product portfolio, with 5G Core being a prime candidate. But the demand is not limited to 5G products. The same requirements exist for cloud-native implementations of the OSS and BSS portfolio.

The vision for CI/CD in telecom networks

The ultimate goal of CI/CD is to allow for automated, repeatable and low-risk updates of all components and layers of the architecture stack. This includes the cloud infrastructure, software applications for the core network and IT domains like OSS and BSS, management of network slices for end-to-end services and the exposure of APIs.

To meet service provider requirements, new releases from suppliers need to be continuously delivered, integrated, verified, validated and deployed. This flow should be fast, and each step automatically triggered by input from the previous one. With updated software made available to service provider repositories, automation will pull new and accepted deliveries into the next environment, facilitating a flow between test and production, or central and national operations.

This will be a journey for the industry where certain parts still need to be defined and developed, but one that is necessary to reap the full benefits of 5G and beyond.

Traditionally, CI/CD automation is implemented to remove manual operational tasks, achieving efficiency and security benefits as a result. Increased efficiency is achieved when engineering and operations streamline their ways of working and manage new services and increased traffic.

Automated handling of updates at more frequent intervals poses lower risk than manual ways of working and large, infrequent upgrade projects. This allows swift, seamless introduction of the latest software release with up-to-date security.

Automation in CI/CD pipelines for network functions (NFs) is based on automated flows provided together with the NF and fits into existing, often standardized, architectures equally as into operational procedures and service providers' ways of working.

With all its significant and proven benefits, this automation remains largely independent of the NF's implementation, either as a VNF or cloud-native application. To unleash the next level of possible gains, the LCM of cloud-native applications must be treated in the cloud-native way.

The CI/CD pipeline

Figure 2 illustrates the CI/CD flow from a service provider perspective. Suppliers continuously make new software releases available, which must then be delivered into the service provider domain. There, they are integrated into larger systems, configured in line with service provider needs and then validated against set specifications. These activities are summarized as continuous integration in the lab, test or staging environments.

Figure 1: Benefits of CI/CD in telecom networks

Increased efficiency

- Zero-touch deployment and testing
- Engineering and operations streamline ways of working
- Manage new services and increased traffic more efficiently

Reduced risk

- Latest software release
- Remove risks that arise from manual ways of working
- Up-to-date security
- Gradual and frequent updates instead of large upgrade projects

Service agility

- Shorter time to market
- Latest capabilities for activation
- Ready to address the diversity of new services

Figure 2: The CI/CD flow for service providers

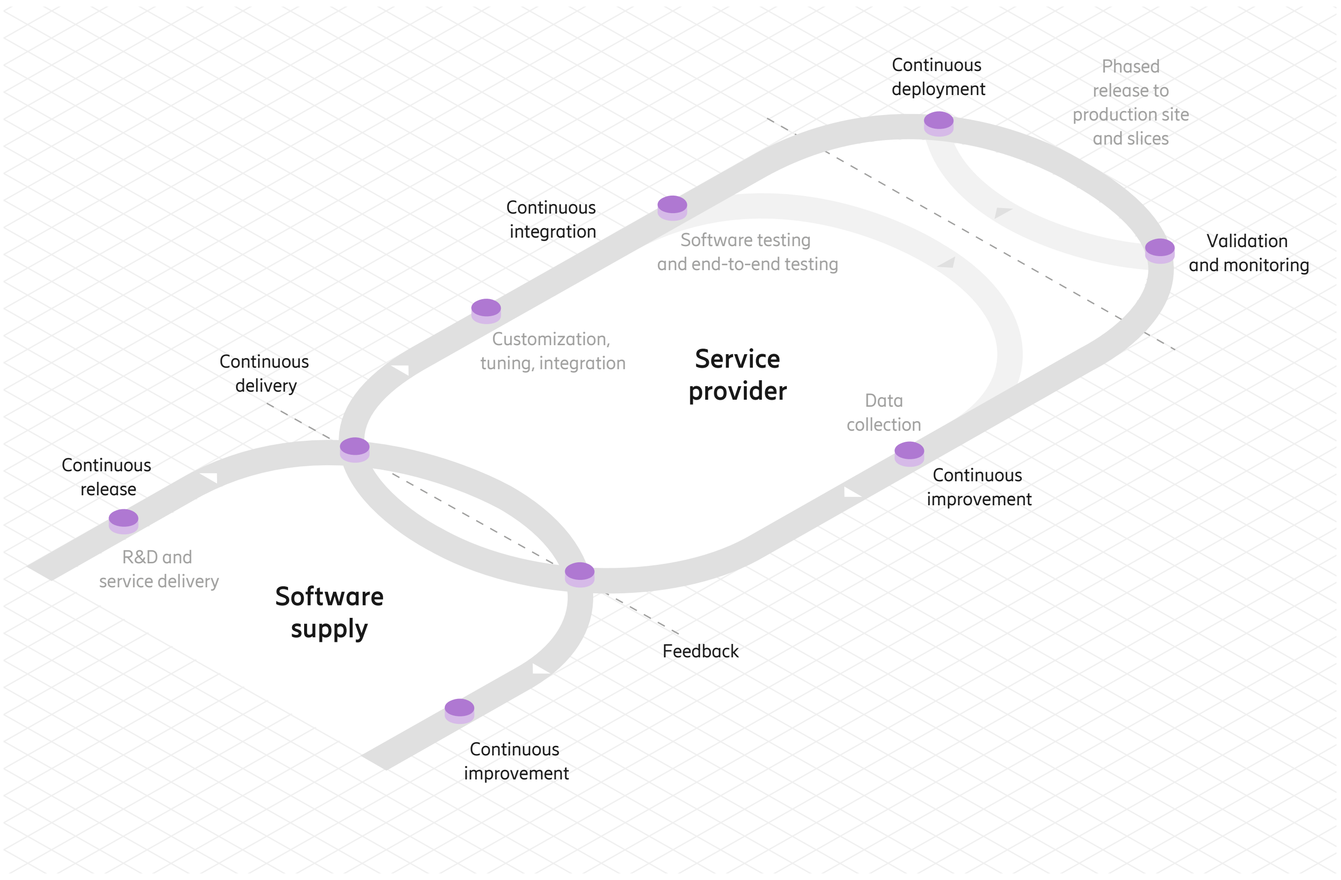
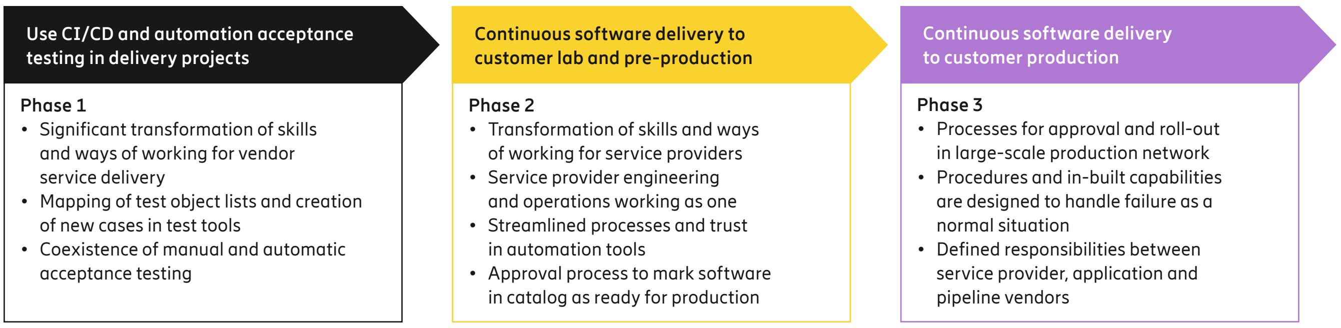


Figure 3: Impact on ways of working as more CI/CD automation is added



Use CI/CD and automation acceptance testing in delivery projects

Phase 1
- Significant transformation of skills and ways of working for vendor service delivery
- Mapping of test object lists and creation of new cases in test tools
- Coexistence of manual and automatic acceptance testing

Continuous software delivery to customer lab and pre-production

Phase 2
- Transformation of skills and ways of working for service providers
- Service provider engineering and operations working as one
- Streamlined processes and trust in automation tools
- Approval process to mark software in catalog as ready for production

Continuous software delivery to customer production

Phase 3
- Processes for approval and roll-out in large-scale production network
- Procedures and in-built capabilities are designed to handle failure as a normal situation
- Defined responsibilities between service provider, application and pipeline vendors

New software releases are either rejected for further improvement or accepted for production at the continuous deployment phase, where they may be upgraded in service, canary tested and then set for normal operation. This process can be executed repetitively across sites.

A key aspect of CI/CD is applying a DevOps mindset and culture, where focus is not only on automation, but also on feedback loops to identify any shortcomings early and allow for correction. The collection of data from the CI/CD flow and the application execution is analyzed through data analytics feeds into both the software supplier's and service provider's continuous improvement processes.

Key steps are executed and tested multiple times in different environments before deployment in a service provider production network, providing software quality assurance and a trusted process.

A phased approach to introduce CI/CD

Moving from physical to virtual NFs was a large internal shift but did not have a major impact on the operational model. With cloud-native applications on the other hand, it is not only about running your code in containers, using Kubernetes or going to a microservices architecture; it will have a major impact on operations and the LCM on all layers of the architecture stack. This transformation is not only about new technology, but is also to a great extent related to new ways of working and developing required skill sets and processes (see Figure 3). We explore the operations aspects in more detail in our guide "Transforming operations on the way to 5G", where CI/CD is an important part.[1]

The time is now for you to further embrace cloud native and start adopting new business processes. This is because disruptive technology enables cloud native and is maturing, its values have been proven in IT and 5G architecture incorporates it. In fact, embracing cloud-native architecture and operations is essential for the secure, efficient evolution of your 5G network.

Moving to a true DevOps mindset

Even though required improvements to operations are clear for most, many service providers recognize that these improvements require a fundamental change to how they design and implement their business processes. For example, it means moving beyond today's ETSI MANO-based orchestration and software management which has been mainstream for a decade, to orchestration in concert with GitOps and Cloud Native Computing Foundation (CNCF) tooling and ecosystems. CI/CD based on cloud-native LCM is now, step-wise, turning into reality, with some service providers taking bolder steps towards its implementation in their networks.

Continuous deployment for cloud-native applications can unleash its full potential when implemented using cloud-native best practices and following the GitOps paradigm. Continuous deployment can then draw on the cloud-native LCM advantages and inherit the benefits of working with version-controlled repositories providing a single source of truth, the possibility to track and manage changes, and enable stable and reproducible rollbacks to any previous version.

The three main cloud-native LCM specific benefits of CI/CD in telecom networks are:
- speed of change:
  - a true continuous delivery model with frequent small updates
  - release corrections and new features much more quickly
  - simple deployment approach for pure software assets
- openness:
  - domain (BSS, OSS, NF) and vendor (inherently multi-vendor) independent approach to LCM
  - separating deployment pipeline technology from orchestration, including hyperscale cloud providers (HCPs)
  - benefit from large CNCF software ecosystems
  - use modern cloud technologies and HCPs
- efficiency:
  - independence between functional and infrastructure LCM
  - orchestration and management to focus on the specifics for telecom (the NFs, the service)
  - replication of configuration for troubleshooting purposes between environments, including from the customer to Ericsson

Cloud-native LCM of applications based on microservices, however, cannot be implemented as an afterthought on any containerized application landscape and OSS management stack. This means you must secure applications, and LCM automation must follow cloud-native best practices with operations adopting suitable ways of working and a DevOps mindset.

[1]For more information, please see the Transforming operations on the way to 5G guide.

Designing 5G applications with automation in mind
CI/CD and LCM for cloud native is a necessary but insufficient enabler for continuous deployment. Prerequisites for successful cloud-native LCM include applications being implemented as good tenants in a CNCF ecosystem (particularly on Kubernetes) and the adherence to existing guidelines for cloud-native applications.

This starts with the 12-factor application principles on microservice level,[2] enabling in-service software upgrade (ISSU) capabilities and evolved configuration management. It requires maintenance of a single code base for an application to be deployed into many different environments, as implemented by Ericsson through One-Track for all our applications, and the Application Development Platform services for code which is shared across our portfolio. In the cloud-native paradigm, an application is composed of microservices with independent life cycles. In CI/CD pipelines, microservices are continuously added and verified into the main product branch. This means each increment is small and can easily be verified or rejected, ensuring quality and robustness of the main software branch.

Following the configuration principle means configuration information is injected into the runtime environment as variables, or as settings defined in an independent configuration file. The benefit of separating configuration settings from application logic is that you can apply configuration according to the deployment path for lab, pre-production, or multiple, maybe differing production environments in local or HCP clouds. LCM of the configuration (separately from the application) in version-controlled repositories is a prerequisite for environments and deployments to be declaratively described and a GitOps way of working to be implemented.

"Disposability" means an application can be started or stopped at a moment's notice. This facilitates fast elastic scaling, rapid deployment of code or configuration changes, and robustness of production deployments. It is necessary for ISSU and cloud-native LCM, where historically bespoke processes for first-time installation, updates, configuration changes or restoration are now implemented as a combination of stopping and instantiation from a source of truth inside the service provider. Rolling updates of the underlying Kubernetes infrastructure rely on the same capability.

What is GitOps?
GitOps is a way of working that takes best practices for development and applies them to software automation. It uses a declarative approach that forms the basis of continuous everything.

The resulting benefits are:
- single source of truth
- possibility to track and manage changes
- stable and reproducible rollbacks to any previous version
- removing complexity by using cloud-native best practices

[2]For further detail, please see 12factor.net.

Where do we recommend service providers start?

All basic, repetitive, time-consuming activities that can be automated must be automated, as they represent the best potential for savings. We recommend that, as a service provider, you start with big-ticket items and quick wins.

As efficiency is gained, new automation needs are identified, making the next steps clear. A typical entry point is the automation of all validation and verification activities. Additional quick wins include software download-related tasks and selected low-level tasks in software preparation and deployment. Examples of possible paths are shown in Figure 4: start with the lab, then move to production (Path 1); automate tasks gradually, then take each to production (Path 2); or a combination of the two.

Individual automated steps can then be orchestrated into larger sections, and finally into a CI/CD flow with less manual intervention.

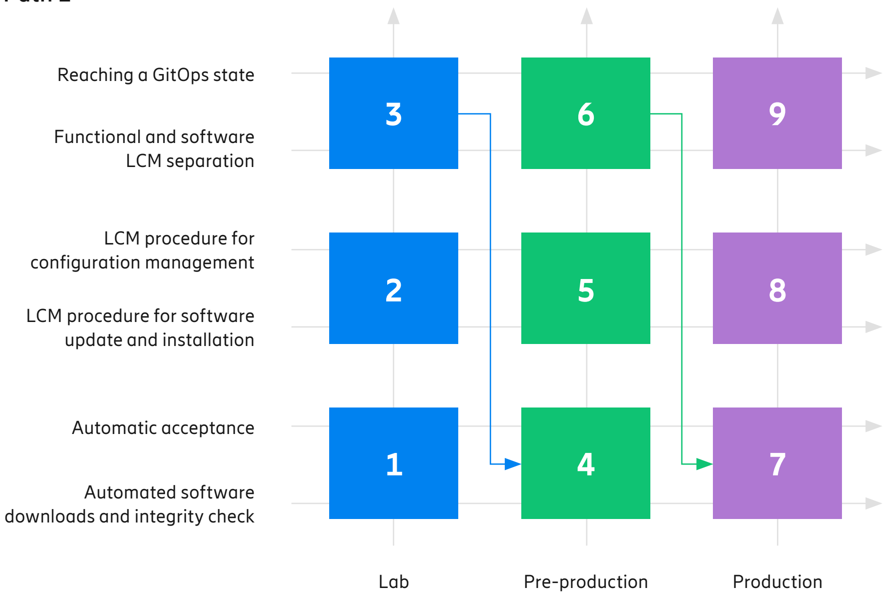
Service providers' needs and approaches to CI/CD will vary depending on their network size and operational and service ambitions. Frontrunners wish to transform their ways of working and operational procedures to become truly agile in the marketplace, while continuously receiving the latest software releases, even at microservice level. They are architecting and implementing CI/CD based on IT best practices to handle LCM of products from their suppliers in a common way.

Conservative service providers would like CI/CD benefits within their existing operational procedures and architecture, for faster product validation and quality at reduced cost. They do not expect a higher release cadence, frequent updates or a continuous software flow.

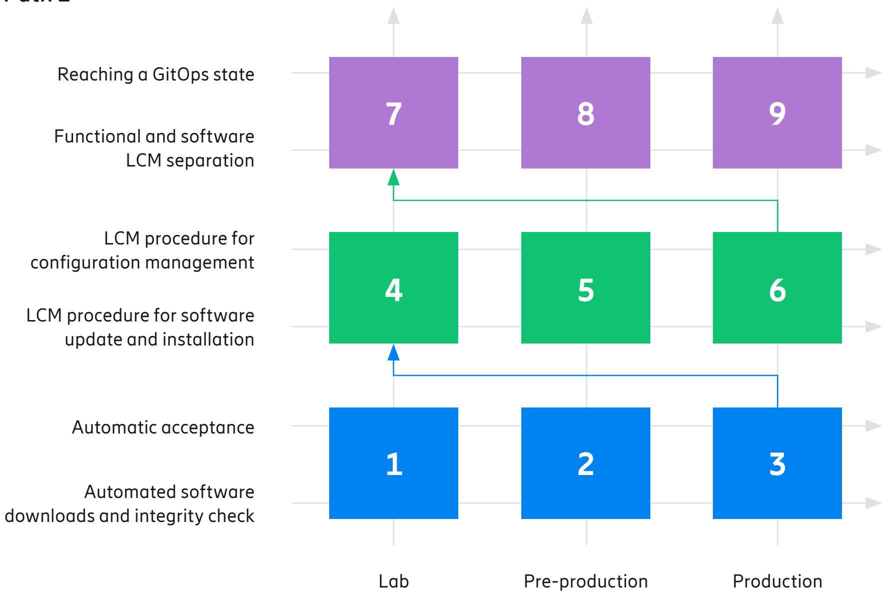
Whichever road is taken, the journey to 5G will require a virtualized network and cloud-native technologies, mandating the selection of applications designed for automation to ease their LCM into a viable CI/CD flow. This signifies a fundamental shift for service providers.

Figure 4: Examples of automation paths to get started

Path 1


Reaching a GitOps state
Functional and software LCM separation
LCM procedure for configuration management
LCM procedure for software update and installation
Automatic acceptance
Automated software downloads and integrity check
3
6
9
2
5
8
1
4
7
Lab
Pre-production
Production

Path 2


Reaching a GitOps state
Functional and software LCM separation
LCM procedure for configuration management
LCM procedure for software update and installation
Automatic acceptance
Automated software downloads and integrity check
7
8
9
4
5
6
1
2
3
Lab
Pre-production
Production

CI/CD with Ericsson

Ericsson's cloud-native products and solutions are evolving towards enabling flexible deployment (as opposed to deployments only based on delivery packaging) and an "everything-as-code" approach to LCM.

LCM is aligned between our products, as our portfolio implements a common set of cloud-native principles referring to agnosticity, state-optimized design, decomposed software, orchestration and automation, and application resiliency.[3] We have developed a unified model which includes tools, processes and methods for cloud-native software design: the Application Development Platform. This specifies the principles for CI/CD, provides and maintains a range of generic microservices for consumption in any product and delivers tools and methods for designing additional, largely independent, microservices.

With the foundation for cloud-native LCM and CI/CD automation in place, you can drive declarative cloud-native LCM into your architectures, and evolve operational procedures and ways of working.

Meeting unified and diverse service provider expectations

If you subscribe to continuous software releases of products, it's natural that you would expect consistent ways of working in your environment. You will also expect that available software is automatically validated and deployed in your lab and production networks.

Ericsson CI/CD implementation based on Continuous Delivery and Deployment (CDD) automatically updates the service provider lab and production networks, independent of product domain or underlying technology. This provides a uniform way of delivering, preparing, deploying and validating software.

CDD and the corresponding automation workflows are developed, tested and released alongside the products. Global usage of CDD in R&D, service delivery and service providers' networks secures quality for each release.

Service provider-specific CI/CD requirements, for example additional test cases and integration of test tools, can be accommodated. Our standard delivery and deployment flows, as provided by each product, can be customized to add functionality or remove individual tasks from the flow.

CDD is based on three main principles:

- It is product-agnostic to scale across the entire portfolio.
- It uses selected de facto industry standards, for example Spinnaker, Ansible and YAML.
- It provides integration points, and builds on tools and products used in the specific target environment.

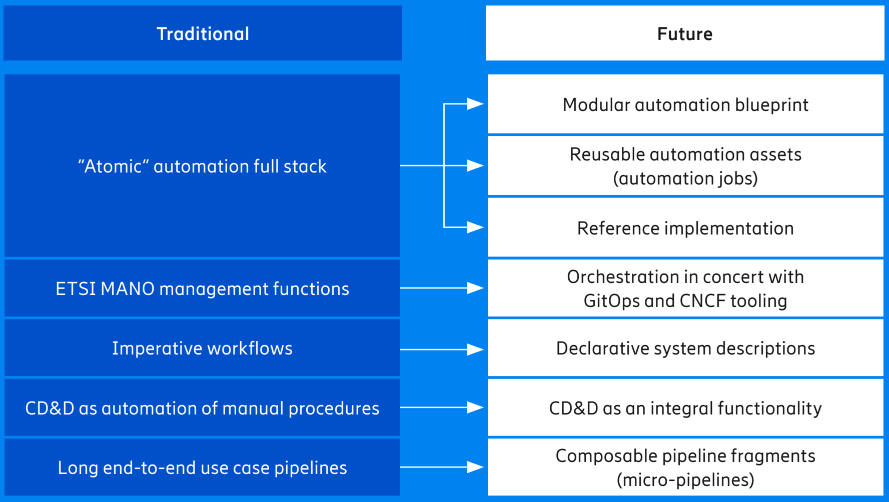
This allows the solution to be used and configured for ETSI MANO-compliant cloud-native network functions (CNFs) and cloud infrastructure, as well as for IT applications that are not managed according to ETSI standards.

All known and unknown products can be supported by providing a framework into which product-specific automation is injected. We achieve this by creating the automation files as part of the design process and delivering these alongside our products, also known as "pipeline-as-code." The automation files are then embedded in CDD in the target environment. Such automation files can also be created offline and injected, allowing the opportunity for products that have not yet implemented the automation files, or for third-party products, to be orchestrated.

For example, in the software upgrade phase, the LCM procedure for a maiden installation or software update is executed, including: pre-update health checks, potential traffic evacuation, the software update and basic post-update health checks.

The software change for an NF is executed by a software LCM manager; our product is the Ericsson Orchestrator and the Evolved VNF Manager.

Figure 5: Simple comparison – traditional vs. future CI/CD


Traditional
Future
"Atomic" automation full stack
Modular automation blueprint
Reusable automation assets (automation jobs)
Reference implementation
ETSI MANO management functions
Orchestration in concert with GitOps and CNCF tooling
Imperative workflows
Declarative system descriptions
CD&D as automation of manual procedures
CD&D as an integral functionality
Long end-to-end use case pipelines
Composable pipeline fragments (micro-pipelines)

[3]For more information, please see the Cloud-native transformation guide.

Accelerating LCM

LCM is increasing in importance on all levels in service providers' networks. LCM of microservices and application instances in any execution environment does not replace LCM on a logically higher NF, network service or system level. In the management, orchestration and assurance domains, focus is shifting from management of the actual instance in an execution environment to securing syntactically and semantically correct representations of the canonical, desired system state versioned in repositories. Its task is to determine the desired state rather than being occupied with the current deployment technology.

Orchestration and assurance on NF, network service or system level has a valuable role to play here. LCM within management and orchestration must, among other things, ensure it does not break a running service through faulty configurations or software updates. Automated measures need to be applied to detect all syntax failures and, as far as possible, semantic failures of configuration files as well as incompatible software versions, to reject or trigger corrective actions when possible before applying change to a system. The management and orchestration suite must take policy and control responsibility. It can then decide what is deployed and where to support the required service and populate or update the runtime repositories.

Automation is evolving from CI/CD being a complementary approach to existing management systems, towards being a more integrated CI/CD approach using declarative system descriptions as a basis. This allows the movement to a modular automation blueprint with reusable automation assets revolving around GitOps as a base technology (see Figure 5). In the automation blueprint, CI/CD tools, automation jobs and GitOps are used in concert with telecom-focused orchestration systems. These systems, based on a network model, determine the desired system state and store it in a Git repository. The updates in the Git repository then trigger small CI/CD pipeline fragments or GitOps agents, which reconcile the desired state with a target system, such as a Kubernetes cluster or a CNF. The modular automation blueprint now allows the composition of various pipeline fragments into larger pipelines, which are being triggered at various stages of the automation and orchestration process.

Our existing CI/CD implementation approach is evolving to create more transparency and tooling choices for LCM of the cloud-native software artifacts within the CNCF ecosystem, while not losing sight of your needs when managing NFs and services or their operational automation needs in lab, pre-production and production settings.

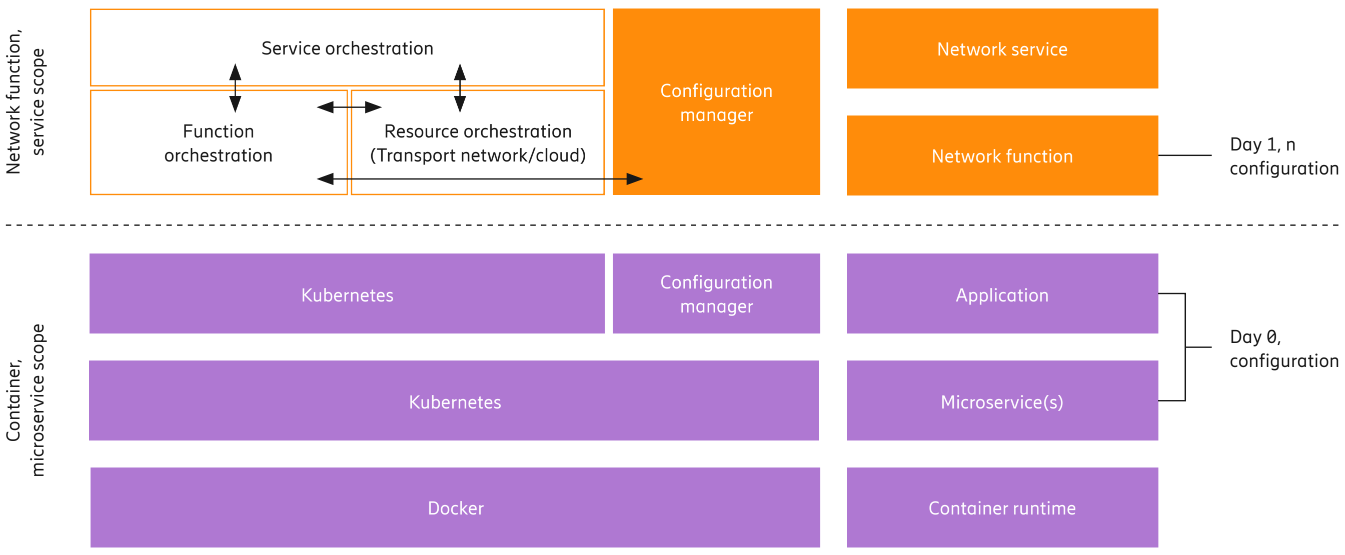
In concrete terms, we clearly separate between LCM of NFs, network-level services and systems with service, function and resource orchestration and deployment pipelines for software LCM based on evolving cloud-native best practices (see Figure 6).

Within the microservice scope, focus is on instantiation and upgrade of microservices, container images and related helm charts including day 0 configuration in any Kubernetes cluster. LCM is implemented through cloud-native pipelines, or through better reconciliation of a described "wanted state" against "running states" in the cluster. The CNCF/IT/HCP ecosystem momentum for deploying technology choices is leveraged, and this limits dependency deployment automation selections in the case of service providers' own implementations or HCP integration.

Pure focus on LCM of microservice and container artifacts facilitates the same principal approach in all stages of the delivery and deployment into Kubernetes clusters, even if Ericsson and service providers take different technology decisions. It enables a common software LCM for all domains (OSS, BSS, NFs) from the implementation perspective.

Running software "gets meaning" and is turned into NFs and services by day-1 and day-n configuration, which is adjusted based on situational awareness, or to meet changing needs. It is our view that network service complexity is solved in the functional domain, leveraging existing capabilities of the implementation domain (cloud native) for execution.

Figure 6: Separation of LCM independence



Within the NF and services scope, the service provider understands the services (business requirements), their intents and has the capability to break these down into the actual resources, applications and their configurations. The translation of network-service intents into configuration of the underlying software can be supported by service, functional and resource orchestration as part of the functional OSS.

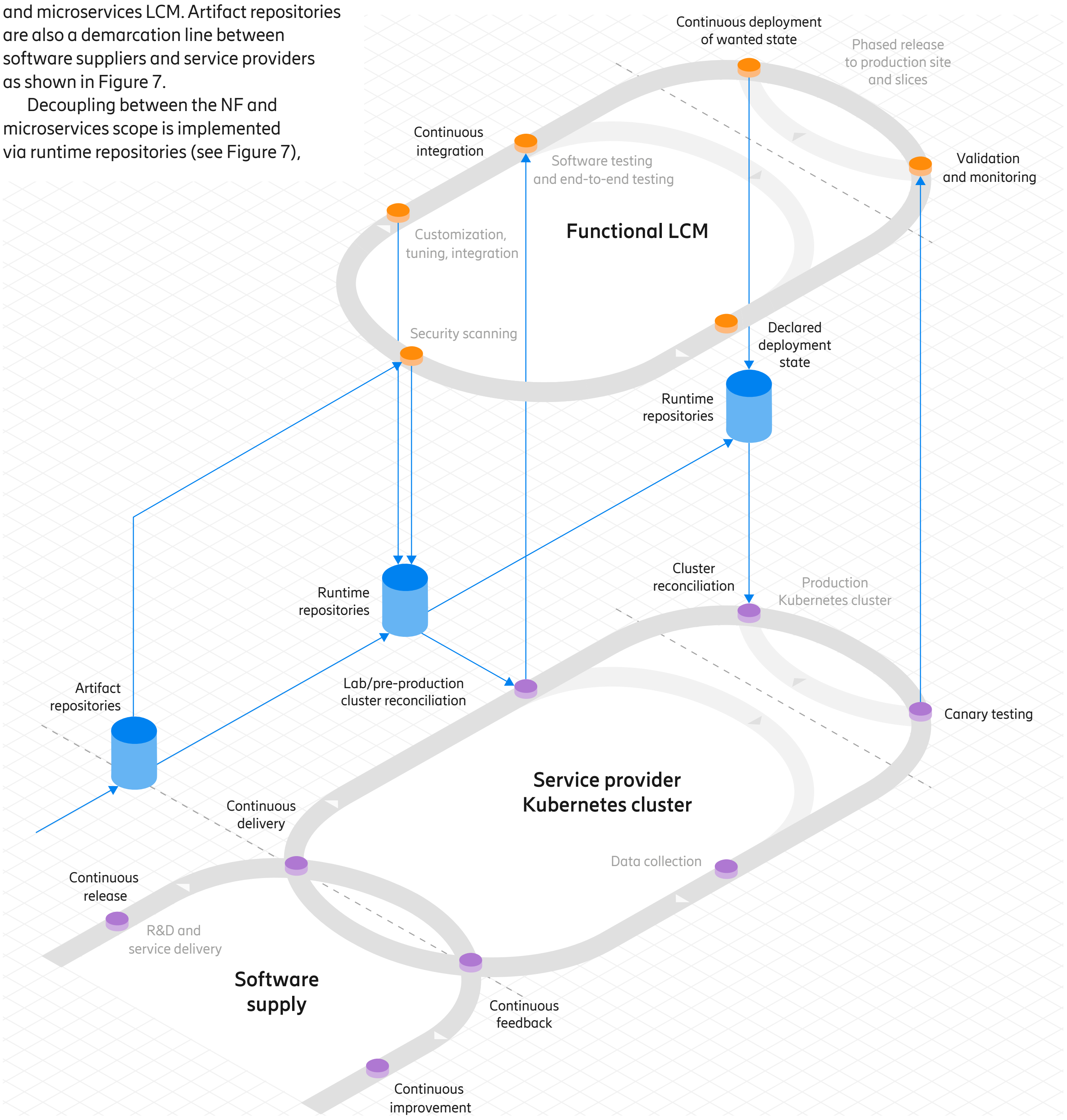
Our experience has shown a clear separation is required between "upper level" functional and service orchestration and "lower level" application and microservice LCM in architectures that implement them both. Runtime repositories are making a responsibility demarcation and decoupling between function and service orchestration and microservices LCM. Artifact repositories are also a demarcation line between software suppliers and service providers as shown in Figure 7.

Decoupling between the NF and microservices scope is implemented via runtime repositories (see Figure 7), which hold the representation of which software should be deployed and where, and which contain the declared state (day 0 differently handled today from day 1, n configuration).

Transparent LCM of the implementation on application/microservice level starts from these version-controlled declarative system descriptions ("as code"). It is within the cloud-native LCM responsibility to resolve the difference between the declared state stored in the runtime repository and the deployed artifacts running in the Kubernetes clusters. This is an application domain or service provider-independent procedure, and should be applicable to cloud-native software from all vendors. Reconciliation towards the target setting is implemented by CNCF ecosystem tooling. With tools like flux evolving, implementing the desired state in a Kubernetes cluster is likely to become the common CNCF way of working.

The CNCF ecosystem is becoming more capable and declarative deployment pipelines (for example, GitOps) are an approach embraced by innovative service providers.

Figure 7: Repositories separating the LCM of functions and software artifacts in a Kubernetes cluster


Continuous deployment of wanted state
Phased release to production site and slices
Continuous integration
Software testing and end-to-end testing
Validation and monitoring
Functional LCM
Customization, tuning, integration
Security scanning
Declared deployment state
Runtime repositories
Runtime repositories
Cluster reconciliation
Production Kubernetes cluster
Artifact repositories
Lab/pre-production cluster reconciliation
Canary testing
Service provider Kubernetes cluster
Continuous delivery
Data collection
Continuous release
R&D and service delivery
Software supply
Continuous feedback
Continuous improvement

From a software-flow perspective, artifact repositories are the landing place for artifacts received from vendors. Service providers' internal onboarding informs artifact consumers (for example orchestration, runtime repository) about availability of new artifacts and optionally triggers their processing. Integration between repositories and control and policy functionality in the management and orchestration domain is needed. With artifacts (and potentially adjusted configurations) being promoted into runtime repositories, cloud-native LCM will reconciliate the cluster and deploy the updates.

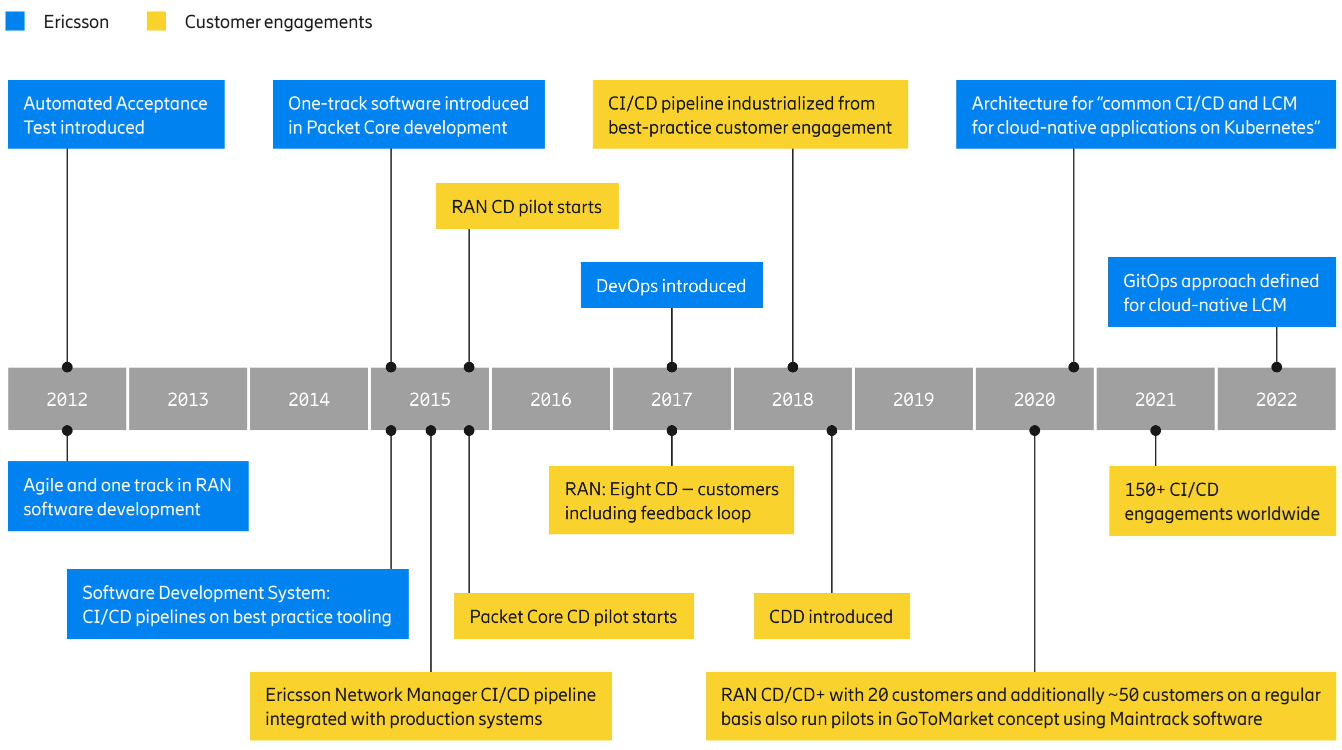
Traditional automation will still be needed, and even evolve. The need for security checks and scans on incoming software artifacts, validation and verification tasks after cluster reconciliation in labs and pre-production will not disappear or be fully replaced through canary deployments. Other customer-specific systems will still need to be triggered. This specific higher-order internal automation improves organizational performance between service provider development, quality assurance and operations. But instead of maintaining large imperative pipelines, these automated flows will be split into micro-pipelines triggered by the service provider or on events occurring within delivery, onboarding and cloud-native LCM.

Based on our experience (see Figure 8), we are evolving our CDD toolset to also handle these events, and trigger event- and task-specific micro-pipelines. These can be provided by us or bespoke for the service provider. Existing Ericsson automation, for example software onboarding or testing, will be able to execute as micro-pipelines and as part of a larger imperative flow as needed.

Key aspects of cloud-native LCM and CI/CD

- Clear separation between the LCM of functions and services and realization management, the LCM of the underlying implementation in terms of applications and microservices.
- Transparent LCM of the implementation of applications/microservices starts from declarative system descriptions ("as code") with the canonical desired system state versioned in repositories.
- Runtime repositories make a responsibility demarcation and decoupling between function and service orchestration and microservices LCM.
- LCM of running microservices and application instances in any execution environment does not replace LCM on a logically higher NF, network service or system level.

Figure 8: Ericsson's CI/CD timeline


Ericsson
Customer engagements
Automated Acceptance Test introduced
One-track software introduced in Packet Core development
CI/CD pipeline industrialized from best-practice customer engagement
Architecture for "common CI/CD and LCM for cloud-native applications on Kubernetes"
RAN CD pilot starts
DevOps introduced
GitOps approach defined for cloud-native LCM
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
Agile and one track in RAN software development
RAN: Eight CD – customers including feedback loop
150+ CI/CD engagements worldwide
Software Development System: CI/CD pipelines on best practice tooling
Packet Core CD pilot starts
CDD introduced
Ericsson Network Manager CI/CD pipeline integrated with production systems
RAN CD/CD+ with 20 customers and additionally ~50 customers on a regular basis also run pilots in GoToMarket concept using Maintrack software

Learnings from global CI/CD deployments

While CI/CD is a new concept for many, we have partners who are already putting these principles into practice, and their key learnings can guide your own CI/CD journey.

Evolution of RAN continuous deployment (CD)
For RAN, the foundation of the CI/CD pipeline started in 2012. It focused on continuous integration and verification, including continuous upgrade capabilities and the automation of test cases in all levels of the development flow. In 2015, T-Mobile US was the first service provider to run a CD pilot in phase 1 (see Figure 3) as a proof of concept, taking regular deliveries from the pre-commercial main track, and carry out release verification in live networks continuously. One year later, CD replaced the earlier release verification setup, and seven service providers, including Swisscom, were onboarded for a long-term partnership with Ericsson, as described in phase 3 (Figure 3), continuously deploying bi-weekly, pre-commercial software into their commercial networks. This replaced the previous release verification setup, which largely corresponds to the continuous integration in Figure 2, thereby collapsing this component to manual or automated software distribution. As of 2022, Ericsson RAN collaborates closely on CD with 22 service providers.

Through continuous automatic data collection from the CD zones, the feedback loop is closed, enabling insights and resolution of issues from the software main track deployed in commercial networks. Close collaboration with CD customers taking pre-commercial deliveries from the RAN software provides critical feedback about both legacy and new functionality and is an integral part of the R&D CI/CD pipeline.

The CD setup serves multiple purposes. Service providers benefit from early access to new functionality, giving them a competitive edge in their markets, and early verification to manage risks associated with service provider-specific configurations in small-scale deployments, minimizing impact on users. We benefit from early and continuous feedback on how the latest functionality behaves in a live environment at limited scale, a form of canary deployment to collect feedback and adjust solutions before releasing to a global audience. The CI/CD pipeline has also proven to be an enabler for more frequent commercial releases. In order to scale the impact of the service provider benefits described above, the intention is to increase the number of RAN CD customers over the coming years. This will also pave the way for service provider operational transformations and automated pipelines required for materializing the full potential of the cloud RAN deployments of the future.

The high number of customers onboard with RAN CD will make it easier for you to introduce cloud RAN where new opportunities with CI/CD concepts described in this paper will be available.

We introduced CD into our network during Q2 2020. It has allowed us to considerably speed up software validation by trying the functionality in advance, reducing the time substantially from commercial release to customer roll-out compared to earlier releases, even while our staff were working remotely due to COVID-19 lockdowns.

Elio Iacovacci,
Head of Mobile Access,
Telecom Italia

CD is key for us to win the network performance benchmarks, and has given us the confidence to deploy RAN software nationwide in just two days. The total effort associated with network upgrades is similar to before, but is now spent earlier and is more evenly spread in time, enabling more efficient planning and the possibility to continuously allocate a small team instead of organizing major, infrequent projects.

Mark Düsener,
Head of Mobile Network,
Swisscom

Core networks

Core networks, and the cloud-native 5G Core in particular, are also domains that we collaborated with service providers on early to explore the CI/CD concept. Building on Packet Core one-track software development and CD pilots in 2015, the journey has continued with a common CI/CD framework addressing the main automation challenges for ETSI MANO-compliant CNFs, as well as for IT applications that are not managed according to ETSI standards.

Learnings from service provider engagements globally show that automation developed and delivered together with a product allows easy adoption. CI/CD pipelines for the preparation, deployment and validation of new software versions is best done by the vendor that designed the software.

Service providers are looking to orchestrate these larger automation flows in a pipeline structure representing their ways of working and organizational setup. This is the most pragmatic approach, allowing service providers to focus their development on what is specific to their environment. Going forward, we observe a demand for even smaller task-specific "micro-pipelines" to give service providers even more flexibility. It facilitates integration with service provider-specific CI/CD technology selection and cloud platforms (HCPs) that require delegated and clearly separated responsibilities between orchestration, products and platform vendors.

In current customer engagements and dialogs, we see a split between service providers continuing an ETSI NFV approach and others driving declarative cloud-native LCM into their architectures, sometimes competing within a service provider. Some are looking at a 2023 implementation horizon. In the latter case, service provider RFQ requirements, open-source activities and internal proof of concepts demonstrate that today's LCM of Ericsson 5G Core on CCD Kubernetes clusters are in line with the architecture described above.

Many service providers we work with have, like us, identified that they need new ways of working and need to change the ways they work with vendors to realize a continuous delivery model.

A lack of trust in software updates/upgrades in the past led to long processes, with many interactions proving hard to automate and efficiency difficult to achieve. While the modelling of existing processes into pipelines seemed a natural first step and helps to reduce impact on organizations, it can prevent operational gains.

Some service providers keep the CI/CD automation processes and tools for cloud-native and legacy software separate. This allows them to build up new teams and ways of working, fully leveraging new technologies without impacting legacy staff and processes. The teams managing the new technologies selectively pick best practices from established ways of working. Using an architecture with micro-pipelines facilitates the selection and reuse of existing best practice. It is important to protect current investments and assure quality for critical infrastructure when moving to cloud native. Software vendors work in a similar way, building new DevOps teams to manage cloud-native products, while leaving the current structure in place for legacy.

The experiences of the first service providers to transition from integrated solution level automation towards LCM on individual products and microservices show that CI/CD is best solved as a combination of orchestration and best practice LCM procedures for software artifacts.

The design of micro-pipelines should be based on open source; however, security, integration and usability must be added on top to allow integration to a service provider production environment. This is an R&D activity requiring understanding of critical infrastructure in the service provider domain and the micro-pipelines should be validated together with the software.

The 5G Core comes on top of containers, which is a new technology that is enabling new ways of delivery. It's about continuous implementation and continuous delivery, and both Vodafone and Ericsson needed to learn how to adapt to a new environment and a new delivery method. We selected one of our data centers and, together with Ericsson, we implemented the 5G Core directly into the live system, which is a brand-new approach. This required a change of mindset in terms of how we bring this new system to life, but also how we bring the new service to life in a live network environment.

Guido Weissbrich, Chief Network Officer, Vodafone Germany[4]

By taking a greenfield approach to deploying its cloud-native dual-mode 5G Core, Telstra had more freedom to try out new ideas with respect to things like cloud-native toolsets, CI/CD pipelines, test automation and continuous deployment activities. With cloud native, Telstra now has the ability to deploy software to deliver new features in a very intuitive manner.

David Aders, Group Owner for Mobile Development & Product Engineering, Telstra[5]

[4]For the full Vodafone Germany case study, please see: Shaping Germany's 5G future.
[5]For the full Telstra case study, please see: Delivering 5G in Australia.

Summary

Embracing automated software pipelines is not optional – it's a must. The more you follow cloud-native best practices and standardized, common ways of working, the closer you'll move to the end goal for all service providers – a zero-touch, automated pipeline.

The software and IT industry continues to evolve at an ever-increasing pace. The telecom industry has been leveraging this momentum and scale by continuously adopting IT approaches. We understand that, as a service provider, you have a persistent need to reduce costs and get new updates, functionality and products to market quicker. This drives the need for an evolution of the approach for CI/CD capabilities, both in terms of technology but more importantly in terms of your processes, ways of working and the development of new skill sets and competences.

This process must be started now. With the implementation of cloud native, our industry will, to a much larger extent, compete for competence with other tech industries, where programs for upskilling the existing workforce will play a crucial role to ensure competence needs are met. We've learned this from our experience collaborating with service providers, for example our work with Singtel on one of the world's first commercial 5G standalone deployments, where their key learnings and insights were characterized as the three Ps: people, processes and platform.[6]

Service providers expect CI/CD to transform its current manual processes, from a process that takes 90 days to introduce software upgrades once, twice, or at most 4 times a year (per NF), into a process that takes 3 days to get an upgrade into production. This will enable the continuous deployment of software across networks. The resulting agility and efficiency benefits will translate into improved customer experience and value creation.

We've identified five key actions service providers must take as their next steps:

1. Separate the functional management from the software LCM, enabling a common approach for software LCM for all domains
2. Evolve the software LCM towards a declarative approach, that can be consistent for different infrastructures including HCPs
3. Evolve orchestration capabilities to leverage the declarative software LCM approach and focus it on functional LCM
4. Evolve the delivery pipeline to beyond monolithic deliveries to smaller artifacts such as microservices
5. Engage in an automatic feedback loop with data from the service provider to Ericsson

Although IT technology has matured to a level where it is applied in telecoms, we still see the need for further alignment in the industry. To facilitate these changes, we are actively participating in different industry forums to improve areas such as:

- creating a consistent approach for how to structure the repositories, reducing integration costs
- forming a technology-agnostic approach to describing the wanted state in the declarative repositories (today it is specific to each technology such as flux)

The implementation of CI/CD is a journey, which has to be addressed in a stepwise approach. It is essential that each service provider takes a holistic approach to their specific journey, based on their strategic priorities. It's clear that the underlying technology plays an important role and is evolving as IT technology is being applied in telecoms. However, the transformation requires much more. It needs development of new processes, ways of working and competencies, to name a few.

It is our strong recommendation that, whatever stage you are at on your journey, you start planning your transformation project today. We are happy to share our experience and learnings from being early to market with leading service providers globally, and look forward to helping you shape your transformation journey and develop impactful strategies.

This guide is part of our cloud-native 5G Core network guide series 2.0, which explores the topics that should be considered when deploying and evolving your 5G Core network.
To discover the full series, please visit ericsson.com/5g-core-guide.

[6]For the full Singtel case study, please see: Transforming Singapore's future with 5G.

About Ericsson

Ericsson enables communications service providers and enterprises to capture the full value of connectivity. The company's portfolio spans the following business areas: Networks, Cloud Software and Services, Enterprise Wireless Solutions, Global Communications Platform, and Technologies and New Businesses. It is designed to help our customers go digital, increase efficiency and find new revenue streams. Ericsson's innovation investments have delivered the benefits of mobility and mobile broadband to billions of people globally. Ericsson stock is listed on Nasdaq Stockholm and on Nasdaq New York. www.ericsson.com

Ericsson
SE-164 80 Stockholm, Sweden
Telephone +46 10 719 0000
www.ericsson.com