FUELING 5G WITH DevOps
DevOps: Fueling the evolution

DevOps has an important role to play in meeting 5G networks’ requirements for faster time to customer in an environment characterized by widely distributed resources and tight constraints on service quality. In collaboration with the open source and academic communities, we have investigated how best to address 5G challenges using DevOps and a generic architecture focused on agility and flexibility.

DevOps approaches extend the agile software development culture to deployment and operations, balancing the development team’s desire for rapid change with the operations team’s desire for stability.

In enterprise environments, DevOps processes and techniques that rely heavily on automation are credited with enabling significant increases in the efficiency of the software delivery cycle all the way into operations. As part of the transition to 5G networks, telecom vendors and operators alike are considering how to adapt DevOps ways of working to boost competitiveness by shortening feature delivery cycles and raising feature hit rates through feedback loops.

5G is expected to deliver unprecedented performance in terms of transmission capacity and packet transit delays, enabling new applications and services in areas as diverse as the Internet of Things, augmented reality and the Industrial Internet [1]. To dynamically define the features supported by the infrastructure and the ways in which these features are managed, 5G networks...
will rely heavily on virtualization technologies [2], as shown in Figure 1.

Network Functions Virtualization (NFV) [2] plays a key role. NFV disaggregates the network function (for example, the router, mobile packet gateway, firewall) from the physical box that contained it. This enables its software implementation to be optimized for deployment on a distributed cloud infrastructure, where an appropriate set of resources may be provisioned dynamically to control resource utilization, energy consumption, and coverage, for example.

DevOps in next-generation telecom networks

The evolution toward virtualization transforms the way both equipment vendors and telecom operators work. Figure 2 illustrates how DevOps can be used to optimize a software delivery cycle, including everything from feature development to operations across disciplines (development, customer engagement and operations) through to continuous delivery (CD) practices. Focusing on automation and lean management practices enables flow control and transparency across the cycle. The organizational and administrative interfaces between the different actors within the 5G ecosystem must be easy to traverse, with appropriate software to secure continuous automation flows.

In a 5G context, the word software refers to both the actual code of virtual network functions (VNFs) and models describing the infrastructure and execution environments hosting this code. While software flows clockwise through the cycle, each stage provides feedback to the previous one (counterclockwise) to allow for software quality improvement and process optimization.

CD practices aim to optimize the flow of software through the software delivery cycle. Through comprehensive, fast and reliable test and deployment automation, it is possible to achieve higher release and deployment frequencies. This leads to shorter time to market and time to customer.

DevOps and continuous everything

DevOps is an interactive approach to product management, development, deployment and operation that stresses communication, collaboration, integration and automation. Working together with the customer every step of the way, the DevOps approach begins with requirement setting and continues through development and operations.

**Continuous integration** – Automated process of secure and frequent integration of source code into source baselines, and binaries into system baselines.

**Continuous delivery** – Automated process of secure and frequent internal provisioning of ready-to-install software product versions of integrated software.

**Continuous release** – Automated process of secure and frequent provisioning of delivered software product to external customers and clients.

**Continuous deployment** – Automated process of secure and frequent production, testing and/or monitoring, and deployment of software products to customer equipment in a live environment.
and it enables improved responsiveness to customer and market demands.

Maintaining one track in software development, using feature flag-driven development, and establishing version-controlled repositories for application code and application and system configuration data enables teams to create a complete environment that is ready for consistent “build and deploy”. Lean management practices aim at process improvement through effective work in process limitations, the monitoring of quality and productivity, as well as the use of application and infrastructure monitoring tools as part of the feedback loop to steer development.

Both CD and lean management practices tie together continuous everything (continuous integration, delivery, release and deployment) activities across teams and stakeholders. When implementing these practices, a number of methods and tools are used on top of an architecture, which provides the capabilities for automation and transparency.

The architecture plays a significant role in building, deploying and operating complex systems. In NFV, it describes how high-level functions typically developed by different teams or open source projects can be interconnected and packaged together to provide a service. Capabilities defined by the NFV MANO architecture [3] allow for dynamic configuration of parameters, dimensioning and scaling a service to reach a wanted set of performance indicators or policies.

The architecture also needs to provide the means for automated monitoring and
Figure 2  Simplified DevOps cycles

- Requirements and production insights
- Development and modeling
- Continuous integration
- Continuous delivery
- Agile operations
- Customer
- Insights and flow control
- Test and validation
- Acceptance
- Continuous deployment
- Continuous release
Figure 3 OPNFV and selected upstream projects mapped on DevOps dimensions
troubleshooting, so that advanced analytics can identify performance deviations from a wanted stage early on, and allow fault isolation in a large system, which eases resolution. Experiences and insights from implementing the automation, optimizing the software delivery cycle and operations are then used to drive the architecture management and improve development and testing of individual functions in a DevOps deployment.

**Telecom-grade open source – a foundation for 5G**

The emergence of NFV technologies has led to a significant increase in the number of open source projects specializing in different components of the NFV stack. A majority of them follow and apply continuous integration (CI) principles and practices to ensure that technical solutions can be developed faster, integrated with other projects and tested in a fully automated way, as well as enabling tailored feedback to developers and users.

However, many open source projects test the components they develop only within their own context without integrating components from other communities. This results in very limited or non-existent end-to-end testing, potentially introducing difficulties when these components are used in a different constellation at a later time.

The open source project Open Platform for NFV (OPNFV) addresses this issue by performing systems integration as an open community effort. Ericsson leads the CI/CD activities within OPNFV, coordinating efforts across different open source communities to ensure the different actors in the NFV ecosystem move toward a DevOps model. OPNFV consumes components of the NFV stack from different upstream projects, integrating and deploying them together, and testing them together in its CI (Figure 3). Like other open source projects, OPNFV applies CI practices strictly. OPNFV brings up and tests the NFV reference platform in a completely automated fashion with no manual intervention, aiming for faster, tailored feedback.

OPNFV strives not to keep any code for NFV components locally in its own source code repositories. When OPNFV identifies issues or missing features, its developers propose blueprints or open bug reports to upstream projects that are then implemented directly in the upstream projects by the same developers. This is enabled by the different feedback loops OPNFV has established. Some of the open source projects OPNFV works with are OpenStack, OpenDaylight, FD.io and KVM.

Since it consumes and integrates components from upstream projects and tests the integrated platform, OPNFV can be defined as a downstream software project. Yet OPNFV also acts as an upstream software project by solving issues and implementing missing features directly in the upstream projects. The combination of the upstream and downstream behaviors therefore makes OPNFV a midstream project.

Due to its midstream nature, OPNFV faces a similar challenge to that of vendors and operators when it comes to integrating the components of the NFV stack to establish a working platform. In order for OPNFV to do the CI successfully, the upstream projects it consumes components from must do CD. Without it, OPNFV will have to wait for official releases rather than having early access to the latest stable versions of those projects. This would greatly limit the OPNFV value proposition by delaying the detection of faults in open source NFV components for months.

OPNFV Cross Community CI (XCI) aims to meet this challenge by providing a production-like environment to its upstream projects. By establishing
the feedback mechanisms between OPNFV and the upstream projects, open source communities are able to assess the maturity of their CI/CD journey, identify their own improvement areas and determine how they can contribute to making OPNFV successful. This experience is highly relevant for both vendors and operators that must integrate software from a variety of sources in 5G deployments.

**Empowering developers in 5G**

Increased development agility and flexibility in 5G will support the traffic growth of the Networked Society and enable new services. To illustrate this, we built a research prototype that targets the integration and delivery of distributed network functions for applications such as industrial robotics and media delivery, spanning multiple administrative and technology domains. We followed the fully automated DevOps pipeline in Figure 2, which addresses every step in the life cycle of an application.

We used common software development toolsets such as Git, GitLab, Advanced Package Tool and Jenkins to build an automated development and delivery pipeline. We integrated the pipeline with the network function life cycle orchestration by implementing software to facilitate the deployment from the development pipeline, life cycle management, policy control, and monitoring capabilities in addition to service and resource governance.

Our orchestration templates were based on several languages: USDL for the service modeling, TOSCA for network slice modeling and YANG for the resource modeling. To support a smooth flow of orchestration logic through the different abstraction layers, we developed transformation functions between the different models. An end-to-end orchestrated industrial robotics application, shown in Figure 4, spans vertically through a business slice, a network slice, a system dimension and a functional dimension. Each slice/dimension has its own representation of the robotics application and its own abstraction of required resources. The application also spans horizontally across several distributed technology and administrative domains.

In the development and integration stage, after initial testing, the code is tagged for packaging into the preferred library format. Libraries are stored in a dedicated repository, and in the integration stage, they are packaged together in the form of a binary software image such as a virtual machine or a Linux container.

Templates for service bundles and basic resource types are defined in the modeling stage. Templates can describe various aggregation levels, from very simple components such as a network function or network service to more complex product level components. For example, in the TOSCA case, templates also describe relationships, topology and life cycle workflows. Templates are stored in a blueprint repository. Multiple templates are aggregated to abstract product-related descriptions that refer to all dependent artifacts and customizable inputs.

Testing and validation are performed repeatedly, coupled tightly with the modeling stage, starting early in the cycle to eliminate errors and improve quality. As described in the TOSCA blueprints, we validate software components and their deployment, focusing on the aggregated types that represent building blocks of complex services. Validated blueprints and related artifacts are tagged as “ready for delivery” during the delivery stage and pushed to production repositories. Validated artifacts can be directly used for product offerings.

The deployment stage spans multiple orchestration levels for an automated end-to-end fulfillment of network services. Deployment artifacts are taken from the product repositories provided in the delivery stage. This flow starts with the uppermost business level, where customer requirements get mapped into the product offerings. Business level mapping is driven by the USDL service model and uses governance, pricing and resource negotiation

---

**SERVICE DEPLOYMENT ON 5G WILL LIKELY REQUIRE THE ORCHESTRATION OF A MULTITUDE OF TECHNOLOGY DOMAINS**
Figure 4 High-level modeling of 5G robotics application use case
inputs. Business service descriptions and Service Level Agreement requirements are further mapped to the network slice requirements and TOSCA blueprint deployment descriptions. The blueprints are customized with dimensioning values and deployment-specific parameters and pushed to the network orchestration layer. The blueprints are then processed by the life cycle manager, which drives the deployment workflow with available resources.

Service deployment on 5G will likely require the orchestration of a multitude of technology domains. Different domains contain specific resources with explicit capabilities and orchestration requirements. A hierarchical layering in the life cycle management makes it possible to cross the technology, functional and administrative domains. In such a hierarchical approach, life cycle management of the individual sub-domains is delegated to the lower orchestration layer. The upper layer handles end-to-end aggregation of sub-domains and higher abstraction-level life cycle management. Isolation properties are defined in the blueprint descriptions.

In the operations stage, a monitoring engine informs the policy engine of any deviation from the required quality level. The policy engine triggers lifecycle workflows designed to maintain the quality of deployed services. Every orchestration layer contributes to maintaining overall service quality with its own optimized workflow.

**Augmented operations capabilities**

Together with 14 European academic and industry partners, we have designed and developed a DevOps framework [4] for efficient deployment and operations of NFV-based services. The results addressed on-the-fly service verification, scalable and programmable observability, and automated troubleshooting.

An elastic router service is a good example of how augmented operations capabilities could be enabled throughout the DevOps pipeline [5]. The service is able to expand or reduce its capacity dynamically, in accordance with customer traffic demand. The elastic router is based on components openly available from upstream controller and virtual switch projects. It realizes elasticity by automatically scaling data plane resources as a function of traffic load.

The various stages in the DevOps life cycle outlined in Figure 2 continuously loop within the respective dimension shown in Figure 3 to provide rapid service agility and dynamicity. A natural entry point to the life cycle of a telecommunications service is the development of the functional service components, such as the elastic router components that provide dynamic scaling of forwarding functionality with centralized control. For this prototype, we used basic development tools such as local integrated development environments with Git/GitLab for code sharing, merging and versioning.

For integration and testing of the entire system (in other words, the complete elastic router VNF), we emulated the network scenario in Mininet, a realistic virtual network environment that is well established in the academic community. In a production system, this would be replaced by utilizing a dedicated

---

**Terms and abbreviations**

- **CD** – continuous delivery
- **CI** – continuous integration
- **DevOps** – a compound of software development and operations
- **GE** – Gigabit Ethernet
- **Git** – version control system for tracking changes in computer files
- **GitLab** – web-based Git repository manager
- **IDM** – Internet Download Manager
- **KVM** – Kernel-based Virtual Machine
- **MHz** – megahertz
- **NAT DHCP** – Network Address Translation Dynamic Host Configuration Protocol
- **NFV** – Network Functions Virtualization
- **OPNFV** – Open Platform for NFV
- **SDN** – software-defined networking
- **TOSCA** – Topology and Orchestration Specification for Cloud Applications
- **UE** – user equipment
- **UI** – user interface
- **USDL** – Unified Service Description Language
- **VNF** – virtual network function
- **WAN** – wide area network
- **XCI** – Cross Community CI
network slice as a sandbox environment for staging, similar to the XCI as offered by OPNFV.

One or more production-ready system components (such as the elastic router VNF) are modeled as a directed graph before releasing and deploying the actual customer service. Such a joint model enables programmability of compute and network resources, making it possible both to define VNF placement and to establish the forwarding overlay in a single transaction.

By integrating new validation capabilities and related interfaces into the release and deployment phases of the customer dimension, service properties (modeled as graphs) can be verified before they are rolled out in the production infrastructure. This is essential in 5G and NFV environments, where reconfiguration of services must be triggered frequently. Misconfiguration of (V)NFs, violation of network policies, or artificial insertion of malicious network functions are a few examples of issues that formal verification methods can identify to ensure service uptime and preserve network integrity and reliability. In our research, we integrated new functions into the delivery, release and deployment components to support continuous, real-time verification of service programming instructions and configurations. These new functions act as gatekeepers, rejecting invalid models or configurations, and providing immediate feedback to the modeling stage. Validated service instances and configurations subsequently passed through the deployment and acceptance stages, which involved the automated mapping (or embedding) of the service model onto infrastructure resources, and the introduction of the relevant service components to the infrastructure.

The agile operations stage starts once services and accompanying network configurations are successfully commissioned. Both services and individual service components must be observed continuously throughout their lifetimes. To support the requirements of future 5G networks, the project team developed and integrated a new software-defined monitoring framework that provides accurate and scalable monitoring both in large-scale, geographically distributed cloud scenarios and in centralized data center scenarios. The framework supports traditional and novel metrics together with local, lightweight analytics functionality, which trigger both local and remote reactions from the orchestration and management architecture in real time.

Agile operations rely on tight interactions between analytics and orchestration workflows. For example, local analytics performed on real-time monitoring data results in production insights, which can be used to support dynamic, autonomous service adaption (scaling) through the direct interface between the VNF and resource orchestration entities (such as Virtual Infrastructure Manager). Here, the service model would be automatically refined to accommodate the new scaling requirement by the VNF itself – that is, controlled by the VNF developer. The DevOps life cycle would remain in the customer dimension, reiterating the modeling stage in an updated service model.

Another example is a local analytics function that would identify problems related to the service functionality that cannot be solved by refining the service model. In this case, these production insights feed into our developer-friendly troubleshooting engine, performing root cause analysis with advanced automation support. Once the faulty service component is identified, the troubleshooting information is fed back to the functional development stage to support debugging and redevelopment of the actual VNF code, thereby closing the DevOps life cycle loop across dimensions.
Conclusion
The stringent requirements of 5G networks are driving the need for further adaptation of existing DevOps practices and toolchains to the telecom industry. Our work with the open source and academic communities demonstrates how to address the 5G challenges related to the evolution of classic telecom fulfillment and assurance processes toward DevOps-powered cycles. Doing so requires an architecture that supports automated deployment and operations, using powerful description languages tailored to different system dimensions that can capture constraints and feature specifications. Transparency of state changes and transitions throughout the architecture enables efficient operations. Our experience in the OPNFV community shows that CD practices including feedback loops throughout the technology stack and across organizations are key to a successful DevOps implementation.

Further reading
- Open Daylight: The Journey to a DevOps Future (September 2016), available at: https://www.youtube.com/watch?v=IILxeD6Kwbs&index=10&list=PLBF5jr6EpGAIrCzJlyboA8DI3_TajTT-2
- Achieving DevOps for NFV Continuous Delivery on Openstack (Verizon case study), available at: https://www.openstack.org/videos/video/achieving-devops-for-nfv-continuous-delivery-on-openstack-verizon-case-study

References:
Catalin Meirosu ◆ joined Ericsson in 2007 and is a master researcher at Ericsson Research in Stockholm, Sweden, where he works on autonomic management for software-defined infrastructure. He received a Ph.D. in telecommunications from University Politehnica of Bucharest, Romania, in 2005.

Wolfgang John ◆ has been a senior research engineer at Ericsson Research in Stockholm, Sweden, since 2011. His current research focuses on novel management approaches for software-defined networking, NFV and cloud environments. He earned a Ph.D. in computer engineering from the Faculty of Electrical Engineering and Computing at the University of Zagreb, Croatia.

Miljenko Opsenica ◆ is a senior researcher at Ericsson Research in Finland who joined Ericsson in 1998. His current research focuses are cloud deployment architectures, orchestration and automation frameworks. He holds an M.Sc. in electrical engineering and computing from the Faculty of Electrical Engineering and Computing at the University of Zagreb, Croatia.

Fatih Degirmenci ◆ joined Ericsson in 2006. He is a principal developer at Ericsson’s Product Development Unit Cloud, where he specializes in automation, CI/CD, DevOps and infrastructure. He holds an M.Sc. in electrical and electronics engineering from Gazi University in Ankara, Turkey, and an M.Sc. in computing from Dublin Institute of Technology in Ireland.

Tomas Mecklin ◆ joined Ericsson in 1993. He is a master researcher at Ericsson Research in Finland (NomadicLab) where he focuses on cloud architecture and related technologies. He holds a B.Eng. in computer science from the Swedish Institute of Technology in Helsinki, Finland.

Torsten Dinsing ◆ joined Ericsson in 2000 and is an expert in service architecture in the CTO office. He is currently a member of the core team driving Ericsson’s DevOps strategy. He holds an M.Sc. in electrical engineering from RWTH Aachen University in Germany.

The authors would like to acknowledge the support received from their colleagues Timo Simanainen, Athanasios Karapantelakis and Róbert Szabó.