ETSI GR NFV-TST 006 V1.1.1 (2020-01)

GROUP REPORT

Network Functions Virtualisation (NFV); Testing; Report on CICD and Devops

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1cec4841-bf9b-4d77-81da-75ea2089807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference

DGR/NFV-TST006

Keywords

CI/CD, DevOps, NFV, NFVI, SDN

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00   Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/3c9c941-bfb-4d7-8da7-5a20890c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2020.
All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE™** are trademarks of ETSI registered for the benefit of its Members and
of the 3GPP Organizational Partners.
oneM2M™ logo is a trademark of ETSI registered for the benefit of its Members and
of the oneM2M Partners.
GSM® and the GSM logo are trademarks registered and owned by the GSM Association.

Contents

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1cc9c941-bf9b-4d77-81da-75ea2089807e/etsi-gr-nfv-tst-006-v1.1.1-2020-01

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1cc9c941-bf9b-4677-81da-75ea2089807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1cc9c941-bf9b-4d77-81da-75ea2089807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1ec9c941-bf9b-4d77-81da-7fea2088080c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

1 Scope

The present document provides guidance and recommendations on how to leverage DevOps and CI/CD techniques across the boundary from SW provider to service provider, or any combination of developer, installation and operational entities. It explores the implications of the processes with regard to the impact of the SW package handoff between SW provider and service provider, the required functionality in the NFV system, the different deployment and operational options by:

- Exploring use cases.
- Defining the steps in the process.
- Defining the metrics and measurements.
- Defining the test procedures.
- Defining post handoffs tests.
- Providing recommendations on VNF Package.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS NFV-REL 006: "Network Functions Virtualisation (NFV); Reliability; Maintaining Service Availability and Continuity Upon Software Modification".

[i.2] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for main concepts in NFV".

[i.3] ETSI GS NFV-IFA 011 (V3.2.1) (2019-04): "Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification".

[i.4] ETSI GS NFV-TST 001 (V1.1.1) (2016-04): "Network Functions Virtualisation (NFV); Pre-deployment Testing; Report on Validation of NFV Environments and Services".

[i.5] ETSI GR NFV-TST 004 (V1.1.2) (2017-07): "Network Functions Virtualisation (NFV); Testing; Guidelines for Test Plan on Path Implementation through NFVI".

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1cc9c941-bf9b-4777-81da-75ea200807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS NFV 003 [i.2] and the following apply:

VNF operator: person or company that operates the VNF

VNF validator: person or company that validates the functionality of a VNF

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.2] and the following apply:

CI	Continuous Integration
DA	Disciplined Agile
ITSM	IT Service Management
JAD	Joint Agile Delivery
SAFe	Scaled Agile Framework
SDLC	Software Development Life Cycle
SRE	Site Reliability Engineering

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1cc9c941-bf9b-4077-91de-75ea2089807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

4 Background on Devops and CI/CD

4.1 Introduction

The basic intention in DevOps/Joint Agile Delivery is to incorporate operational and other related aspects of the entire software lifecycle as early in the development process as practical, and to ensure smooth delivery jointly across organizations. This requires that many steps in the process are more continuous, incremental, and iterative than in traditional waterfall models, and higher levels of automation are applied to the delivery pipeline, including testing.

To that end, there are a number of processes and concepts involved; these are described in the following clauses of the present document.

The context of the present document is the testing elements of DevOps/CI/CD; the present document does not seek to provide a comprehensive overview of these topics beyond that scope.

4.2 General Backgrounder

DevOps is a combination of practices that embody a culture of end-to-end ownership and shared responsibility across teams from Development, QA, Security, Operations, and other areas as a single cohesive, service focused team. DevOps takes much of its heritage from the areas of lean manufacturing and operational management theory. This is about applying industrial design techniques to the software industry.

Joint Agile Delivery (JAD) is a series of multi-party interlocks to support agile software development and delivery in a complex carrier environment. It does this through the expected liaison at key points along the way, such as around requirements, testing, and acceptance. It leverages CI/CD tools and techniques to ensure robust, repeatable, rapid delivery of each code iteration. It is compatible with agile software delivery methodologies such as the Scaled Agile Framework (SAFe), and Disciplined Agile (DA).

As hardware becomes commodity, a new technology base of cloud and virtualisation, microservices, and APIs has emerged. These technologies have provided an inflection point - an opportunity to re-examine the way in which ICT is performed and managed; to distil key processes down to their essence, and re-apply those principles in a way that leverages these technologies to provide a better way to manage quality and change. This new approach has dramatically altered the risk landscape, and this has an impact on the operating model, if an enterprise is to recognize the benefits of this new approach.

Some of the most important aspects of this change include:

- **Cloud versus bespoke infrastructure solutions**. Traditional applications required bespoke infrastructure solutions. Often the production environment was, due to cost, significantly different from other environments. This had implications in how it was maintained: managed in-situ across an extended lifespan; all-or-nothing changes executed in ‘quiet' times, often involving service disruption; configuration divergence between environments from development to test to production having implications on testing and quality assurance. Cloud, by contrast, has the potential to eliminate all of these concerns.

- **Infrastructure as code**. The nature of cloud environments is that infrastructure can be created, modified, and deleted using APIs. This means that it can be created under programmatic control. This implies: repeatability and consistency, eliminating human error, and elimination of the entropy associated with having to maintain bespoke, manually configured infrastructure in-situ across an extended life span.

- **All environments virtualised**. The virtualisation of all environments provides an opportunity to ensure consistency across environments never previously available (or cost prohibitive). This significantly reduces the risk associated with change, both due to consistency of the environments, and consistency of the change process used to promote code between environments.

- **Microservices; elimination of heavy integration overheads**. A significant source of enterprise risk is the integration of a large number of changes rolled up into major releases. This leads to extended integration timeframes, and a lack of direct traceability to the source of errors found during this integration cycle. The industry trend towards discrete microservices directly addresses this risk, increasing both velocity and quality simultaneously. The limited scope of each service makes testing more predictable and simpler. The use of APIs as the entry point protects consumers from internal changes, further simplifying integration testing. Each service can evolve at its own speed, no longer enforcing large, complex integration cycles.

Against this technology backdrop, DevOps institutes a set of cohesive software delivery practices:

- **Continuous Integration**. The defining characteristic of DevOps projects is their creation and ubiquitous use of a delivery pipeline which controls the advancement of release artefacts from idea through to production. This is not a static pipeline, nor is there only one. The pipeline more closely resembles a factory production line with many conveyor belts and components being assembled until ultimately brought together to make a new car. Each of these components have their own pipeline and operate under their own rules and speed.

- **Small, discrete changes**. A significant source of work and error is the delayed integration of new features and code changes into the "baseline". **Continuous Integration** espouses a view that work is continuously checked into the baseline, to catch any integration related issues early, and ensure that the product is always stable and passes regression tests. This is then coupled with **Continuous Delivery** to ensure that a new, clean version of the product is always shippable. This combination means that there is a confidence in the build and release process that supports a significantly faster end-to-end release process, further creating a virtuous cycle of quality and velocity.

- **Automated testing**. Supporting continuous integration and the rapid promotion of changes through the delivery pipeline, automation of testing is held out as an essential aspect of DevOps. Not all testing can be automated, but automation should always be sought to the extent possible.

- **High degree of automation**. Beyond testing, automation is applied to all steps in the delivery pipeline, including generation of intermediate artefacts, packaging, environment creation, delivery, deployment and event response. This significantly increases both the power and authority of the development team.

- **Monitoring, metrics, and real world feedback**. DevOps is about rapid and continuous feedback. A key focus is on instrumentation and operational telemetry, and using real world feedback rather than relying on requirements and test environments.

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/4ce0c941-bf9b-4d77-81da-75ea2089807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

- **Cohesive, cross-functional teams; shared responsibilities**. One of the most talked about elements of DevOps is the integration of previously siloed expertise into a single, cohesive team with shared incentives and responsibilities.

Beyond these base practices, a number of "industry best" practices are also often cited:

- **Version Control and the "Left Edge"**. Critical to successful DevOps projects is the use of version control over all **source artefacts** (not just source code). This applies to less obvious source artefacts, such as visual workflows and related design artefacts. Developers only have write access to the "left edge" - the source artefacts. Everything generated from those artefacts is designated an **intermediate artefact**, that cannot be directly altered, and is re-generated by committing new source changes. This ensures that version control is enforced, and that every version of the system, and every element of the system is reproducible.

- **Continuous Integration Triggers and Automated Testing**. Another aspect of a mature CI delivery pipeline is the implementation of triggers - that the pipeline CI controller can detect events and automatically take action, such as continue to promote the object along the delivery pipeline, without human intervention. The most obvious example of this is where a code check-in causes the automated test suite to be run, in order to determine if the build is still clean after the most recent code change. CI triggers include source code repository triggers (code commit), test triggers (success/failure), and human approval triggers (approval to promote).

- **Use of "ephemeral" infrastructure**. A major historical source of error was the entropy introduced by the in-situ maintenance of infrastructure over an extended period. A cloud best practice is to generate new infrastructure for each new release, build and release the new version of the service onto that infrastructure, then to cutover/migrate the workloads to this new version in a controlled manner, and eventually destroy the virtual infrastructure of the previous version. This means that the current and previous version of the service remain operating in parallel until the migration is complete, and this provides additional mechanisms for managing risk.

- **No rollback, only fail forward**. The in-situ nature of traditional large systems meant that change was "all or nothing", and if a change failed, it would be "rolled back" and the system returned to its previous (pre-change) state. This technique was always fraught, as any transactions applied from the time of the change would also need to be rewound and then reapplied, something that was not always possible. It would also often mean restoring from backups - a procedure often untested except in high risk circumstances such as restoration during a major event; complications would abound. In mature DevOps environments, a bug is addressed at the source (left edge) and the tests that allowed the bug to slip through are also addressed. The velocity of the end-to-end process making it faster and more reliable to push a new version quickly. Where necessary, functionality might be immediately disabled without any new code change or rollback (see next).

iTeh STANDARD PREVIEW
(standards.iteh.ai)
Full standard:
https://standards.iteh.ai/catalog/standards/sist/1ce0c941-bf9b-4d77-81da-75ea2689807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

- **Decoupling of Deployment from Activation**. Advanced DevOps practices include the use of "feature flags" to decouple deployment from activation. This addresses the most significant risk associated with change - the all-or-nothing nature of a release. This technique has service code consult a dynamic configuration service to determine whether a feature should remain dormant, be activated for a particular set of users (e.g. specific users, specific locations, specific volumes, etc.), or be fully activated for all, and this becomes the basic risk control mechanism to control the potential impact of a new feature or release.

- **Continuous Deployment**. With deployment and activation decoupled, delivery pipelines can extend all the way into production. Continuous Deployment, sometimes called "push on green" is instituted where sufficient confidence exists in the delivery pipeline, including the automated testing steps.

- **Continuous feedback, A/B testing**. All of the above features provide an environment of continuous feedback, where real production data is used to guide development and operations practices and priorities. Feature flags can be used to present alternate versions of a feature to different users, providing a way to compare their usefulness and value (A/B testing). This moves software development from the guessing associated with traditional requirements to the real world feedback of how the system is used.

- **Site Reliability Engineering (SRE)**. A sister movement to DevOps, SRE is the application of IT Service Management (ITSM) at cloud scale. Pioneered by Google^{TM}, Amazon^{TM}, Netflix^{TM} and other Silicon Valley^{TM} cloud companies, SRE is the application of automation to ICT operations, and the evolution of the IT operations function.

4.3 DevOps in an NFV Context

In the context of Network Function Virtualisation, all of the above benefits can be brought into play. Smaller, incremental releases of VNFs and VNFCs can be rapidly promoted through a continuous integration/delivery pipeline as described in the present document. The use of discrete functions and APIs help to promote rapid evolution and delivery of services in a complex, multi-vendor environment.

To recognize the full benefits of DevOps, significant process retooling may be required. The techniques described in the present document can help form a basis for planning such activities.

4.4 DevOps in a Multi-Party Setting (Joint Agile Delivery)

A further complication for NFV operators is the coordination and alignment of efforts across organizational boundaries. The operator may make use of functions from one or more suppliers. Each of these organizations may have their own DevOps Delivery Pipelines, and effort is needed to coordinate and synchronize across the lifecycle to ensure the value from DevOps practices is realized. "Joint Agile Delivery", as described in the present document, provides a framework for this coordination between parties, ensuring the benefits of DevOps in the complexities of an operator environment.

4.5 Delivery Pipeline

The "delivery pipeline" describes the end-to-end process, from idea to production. This pipeline is modelled on the traditional Software Development Life Cycle (SDLC) as a baseline, including SDLC activities such as requirements capture, design, coding, test, and release, but applied at a more fine-grained resolution, following an iterative software development methodology such as agile software development. Further, DevOps views this pipeline as a production line of activities that should be automated to the extent practical.

As software components, VNFs and VNFCs are developed, unit tested, and run in one or more test, staging or pre-production environments before they are deployed into a production environment. This approach allows development, operations, security, and other teams - in both the supplier and operator organizations - to build confidence in the release through testing, and to support a joint agile VNF development and delivery process all the way through to production deployment and operation.

iTeh STANDARD PREVIEW (standards.iteh.ai) Full standard: https://standards.iteh.ai/catalog/standards/sist/1cc94b6b-4077-810a-75ea2089807c/etsi-gr-nfv-tst-006-v1.1.1-2020-01

The pipeline steps can be all within a single organization, or split across organizations. There may be different additional steps involved when a cross-organizational model is applied. For example, the testing environment might be split into test of the total VNF, a single VNFC, or a network service. Some operators might share environments with suppliers to assist with integration testing, others may force a hand-off between environments for this purpose.

Naturally, this introduces its own set of challenges, since the process should be reliable. For telecommunication environments, several check-points are likely to exist where a software component may not progress further down the line.

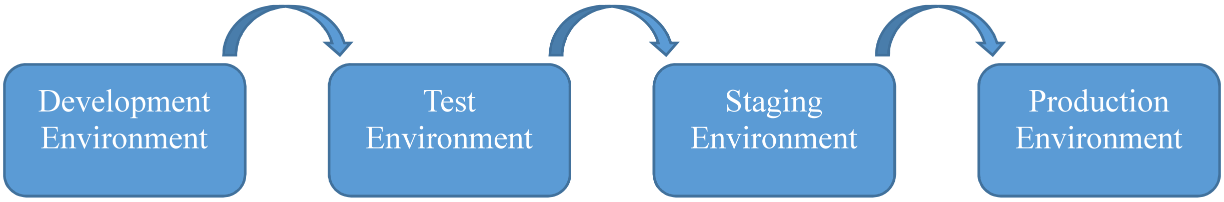

Figure 4.1: Example Delivery Pipeline (Baseline version)