ICT Project Guidance

Development: Delivery Environments

Version:

0.2

Author:

Sky Sigal, Solution Architect

## Description

This document introduces to non-DevOps engineers a list of environments -- along with a high-level explanation of their use – that common to a DevOps pipeline managed delivery environment.

## Synopsis

DevOps relies on multiple distinct environments built from infrastructure as code instructions. This is to permit automation to work without conflicting with environments that delivery stakeholders are accessing at the same time.

## Contents

[Description 1](#_Toc151979812)

[Synopsis 1](#_Toc151979813)

[Contents 2](#_Toc151979814)

[Introduction 3](#_Toc151979815)

[Issue 3](#_Toc151979816)

[Risks 3](#_Toc151979817)

[Resolution 3](#_Toc151979818)

[Environments 4](#_Toc151979819)

[Non-Production Data Environments 4](#_Toc151979820)

[Built Test (BT[-xxxx]) 4](#_Toc151979821)

[Developers Test (DT-xxxx]) 4](#_Toc151979822)

[System Test (ST[-xxxx]) 4](#_Toc151979823)

[User Test (ST[-xxxx]) 5](#_Toc151979824)

[Interoperability Test [IT-xxxx] 5](#_Toc151979825)

[Training (ST[-xxxx]) 5](#_Toc151979826)

[Pre-Production (PP) 6](#_Toc151979827)

[Non-Production Data Backups Environment 6](#_Toc151979828)

[Production Data environments 6](#_Toc151979829)

[Production (PROD or PR) 7](#_Toc151979830)

[Production Data Backups Environment 7](#_Toc151979831)

[Other Considerations 7](#_Toc151979832)

[Identical Environments 7](#_Toc151979833)

[Brach Numbing 7](#_Toc151979834)

[Conclusion 8](#_Toc151979835)

[Appendices 9](#_Toc151979836)

[Appendix A - Document Information 9](#_Toc151979837)

[Versions 9](#_Toc151979838)

[Images 9](#_Toc151979839)

[Tables 9](#_Toc151979840)

[References 9](#_Toc151979841)

[Review Distribution 9](#_Toc151979842)

[Audience 9](#_Toc151979843)

[Structure 9](#_Toc151979844)

[Diagrams 9](#_Toc151979845)

[Terms 10](#_Toc151979846)

## Introduction

A mature delivery of IT services is traditionally done via deploying in sequence to multiple distinct environments for approval by various groups of stakeholders (e.g., developers, then testers, business users, etc.).

The sequence used in different organisations may vary from a very rudimentary sequence of only two environments (e.g., only a Non-Prod and Prod environment) to something slightly more complex (e.g.: System Test (ST), User Test (UT), Pre-Prod (PP) and Production (PROD).

These environments were traditionally set up by hand and later modified as needed, one by one, sometime remembering to document the changes, sometimes not.

## Issue

The manual development of environments, combined with incomplete documentation, leads to predicable surprises and issues at crucial times, in turn leading to a reactionary response of requiring more manual, repetitious testing to try to mitigate these issues. This increase in effort in turn impacts cost and deployment, leading deployments to be further and further spaced apart, delivering changes later and later to end users.

## Risks

The use of delivery stakeholders with more experience in manual deployment and manual testing risk constraining or even undoing the benefits and value proposition of the automated approach required to implement DevOps, Automation and Cloud development patterns.

## Resolution

This document provides a high-level introduction to environment types that are common to DevOps practices, and what they intended for, by whom.

# Environments

## Non-Production Data Environments

### Built Test (BT[-xxxx])

The first environment that a newly compiled and packaged system is deployed to is not used by people – it’s an environment that automation can use to run dynamic integration tests as a step within the deployment test.

A deployment that fails its dynamic tests in the Build Test environment is no progressed further and therefore doesn’t get deployed to a Developers Test environment.

### Developers Test (DT-xxxx])

If the pipeline has successfully run dynamic tests on a deployed package within the BT environment, the pipeline redeploys the package to a Developers Test environment.

As the name implies, the Developer Test (DT) environment is built for Developers, for them to test the system before it progresses to others to test. It is shared amongst developers for them to be able to log on and visually confirm that functionality is running as expected.

Note:  
In practice, the DT environment is not used very much – but it is a useful way to quickly confirm that packages can run on environments other than a developer’s station.

### System Test (ST[-xxxx])

The System Test is an environment built for Analysts.

In the past -- before pipeline automation became widely available, approximately 2010 onwards – System Test was used by Analysts to manually automate test scripts and visually confirm results.

This approach is now considered poor practice[[1]](#footnote-2). Tests should not be performed manually by people. Tests should instead only be performed by automation, as part of the dynamic tests stage in a pipeline.

Hence this environment is used by analysts to do exploratory work to guide the defining of new Acceptance Criteria as parts of new Work Items.

New work items are developed into dynamic tests by developers.

Note:  
Retraining manual testers into no-code/low-code developers has dubious merit in terms of quality and maintainability of the outcomes[[2]](#footnote-3) – whereas if reserved to focusing on what they are highly trained at, i.e. developing acceptance criteria, is considered significantly higher value.

### User Test (ST[-xxxx])

The User Test environment is an environment stood up for Subject Matter Experts representing Users can access the system, to be involved, see progress, manually explore the system, provide feedback if required, and/or give confirmation that the outcome meets expectations.

Note:  
Although the name implies that UT environments are for access by end users, they are not. They are stood up for access by Subject Matter Experts representing Users.

**Considerations:**

* UT environments authenticate using Test Domain – *not* Production – domain Identity providers (e.g., the *OurOrgST* domain’s AAD, not the *OurOrg* domain AAD).

### Interoperability Test [IT-xxxx]

An environment is stood up to permit external systems connect to the system’s APIs, authenticating themselves by OAuth 2.0 or similar. The environment is not for access by end users.

**Consideration:**

While the environment uses non-production data, the environment should be authenticated using the Prod directory service (e.g., *MyOrg*) so that they are using the same account they will be using in the Production environment discussed last.

### Training (ST[-xxxx])

If a system has published documentation sufficient for users to self-help and/or provided an assisted support channel, training may not be required. In many cases, it still will be.

An environment that can be regularly reset is required for this.

**Consideration:**

* A specific set of data, for trainees to follow a training script, would be useful.
* While the environment uses non-production data, the environment should be authenticated using the Prod directory service (e.g., *OurOrg*) so that they are using the same account they will be using in the Production environment discussed last.

### Pre-Production (PP)

The pre-production environment is an environment built for Maintenance stakeholders.

It is customary to deploy to an environment before production, to ensure correct integration, using production environment credentials.

Note that this practice is relatively moot, due to deployments to the environments listed above:

* The BT environment has run automated tests that demonstrate integration works as intended.
* Both the Interoperability Test and Training Environments demonstrates that integration has been successful using Production domain credential.

Note:  
The only credentials that have not been demonstrated are those used to connect to devices containing production data. But this would be considered both insecure and potentially illegal as this use would probably not be disclosed to end users.

**Considerations:**

* Consider using a secondary, non-production, database on the same data store. This would enable checking for firewall rules and other potential integration issues, while not using production data.

### Non-Production Data Backups Environment

The data of the above non-production environments is backed up and/or restored from backups persisted in a non-production environment. The process of backup and restoration should be done completely by automation.

Note:  
Due to cost, it is common that one device is shared amongst all the environments, with automation used to follow a naming convention that disambiguates each database by environment and branch number.

## Production Data environments

There are only two environments that have production data in it.

### Production (PROD or PR)

All the previously described environments use non-production data. Of note, If the data was ever inappropriately accessed, compromised, or released, no harm to end users would occur.

The Prod data is different. The data it manages is that of end users who if any of the above were to occur, could be harmed.

Note:  
Production data, containing Personal Information (and Personal Identifiable Information, can **only** be collected for use legally in the manner it was disclosed to end users to which they have agreed.

### Production Data Backups Environment

This is an environment that is separate from the previously described Prod environment, to which production data is backed up. Data is *not queried or accessed or modified here* (hence does not require being disclosed). It is only here for access by automation to restore data to the Prod environment.

## Other Considerations

### Identical Environments

In the past, system maintainer specialists developed environments. In the era of DevOps this approach is not recommended. Instead, the specialists develop code instructions to develop environments.

This code is run by automation as an early step in a deployment pipeline.

The code is “Infrastructure as Code”, relying on managed 3rd party services, rather than devices (even if this implied by the term “Infrastructure” in IaC).

The outcome is that each environment will be identical – the only thing changing in between them should be the credentials used for integrations – further evidencing that the system is instantiated as intended way before it is deployed to the Prod environment.

### Brach Numbing

In the past, when the cost of hardware and the effort of developing environments was manual and therefore expensive it was expedient to develop the fewest environments.

Cloud based infrastructure – when used as intended -- is considered much cheaper, and it is now best practice to automate the development of a line of environments – BT, DT, ST, UT, IT, PP, PR -- *per branch*. To disambiguate between branches, the environments are suffixed with the branch number.

Note:  
The only branch that does not follow this convention is the default/trunk branch – the branch number is omitted.

## Conclusion

The delivery and resource cost of DevOps cannot be delivered as expected without reliance on automation.

The use of automation -- instead of manual operations – in turn requires a change in the processes that cannot be automated, so as to not conflict with the automated operations.

Also required is a small change to the full of environments for automation to work on without interference and or conflict with operations by delivery stakeholders (e.g.: the BT environment).

Finally, the manner of creating environments requires change, from manual operations that are not well documented, to automated code instructions that can destroy, create, and/or rebuild environments in a predictable, repeatable way while ensuring that data is backed up and restorable so that data is not lost.

Appendices

Appendix A - Document Information

### Versions

* 1. Initial Draft
  2. Minor Changes

### Images

### Tables

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
| Sandy Britain, Enterprise Architect |  |
| Matt Duguid, DevOps Specialist |  |
| Amy Orr, Data Domain Architect |  |
| Russell Chapman, Project Manager |  |
| Rodney Snell, Business & Tech Lead |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Terms

Refer to the project’s Glossary.

##### IT

: acronym for Information, using Technology to automate and facilitate its management.

##### ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. IT is a subset of ICT.

##### DevOps

: a software development, delivery and operations methodology that relies heavily on the automation of infrastructure development, packaging, deploying, configuring, provisioning, and testing.

1. more often than not it produces perverse outcomes -- including unsafe and illegal ones -- such as demanding to have access to integration credentials as well as often working with some form of production data in non-production environments. [↑](#footnote-ref-2)
2. It takes approximately 7+ years to develop a novice – which is where testers start -- into a senior quality developer. [↑](#footnote-ref-3)