Published draft

OPUSVL

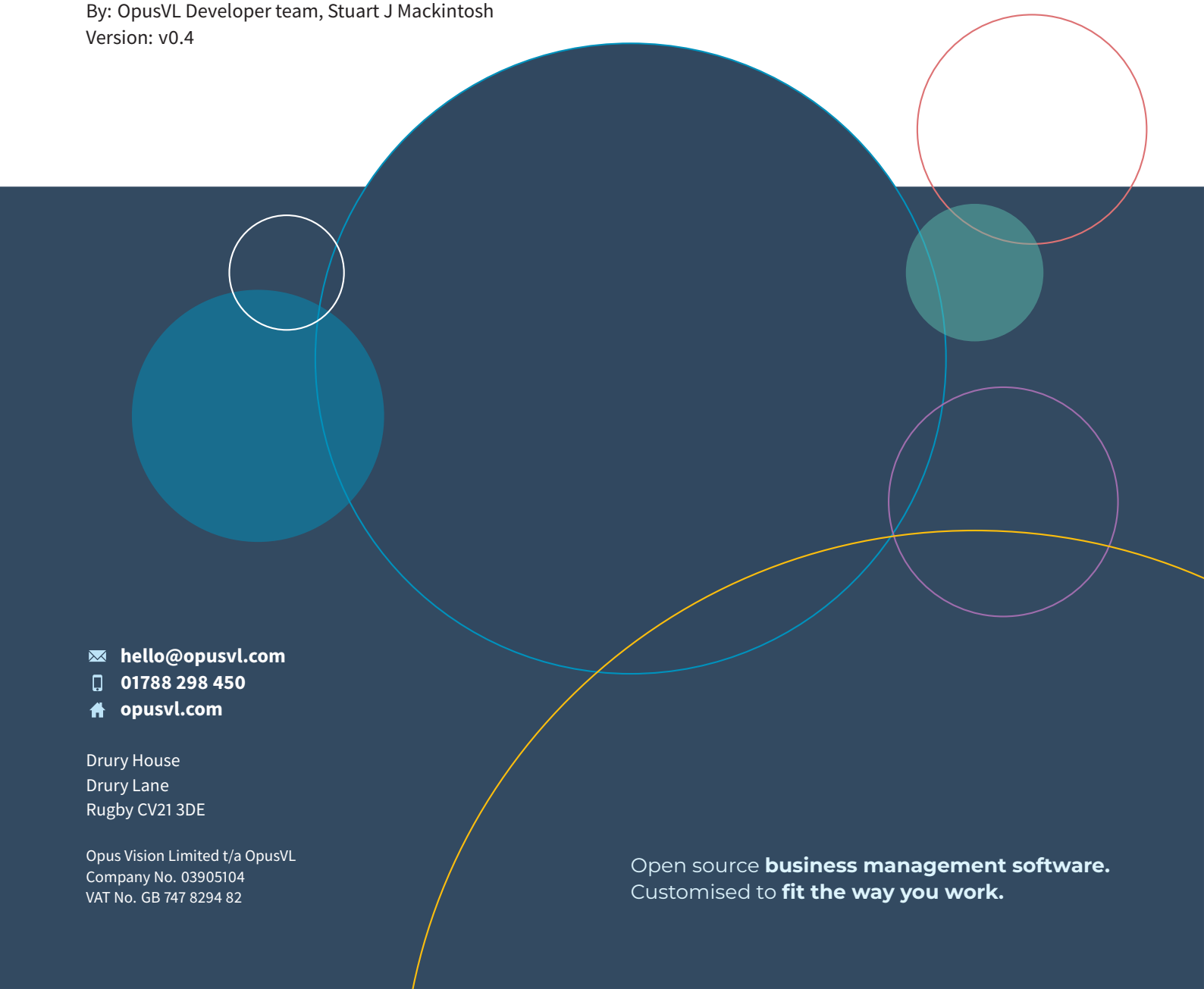
Document overview

- Release pipeline overview
- Deployment paths
- Operational environments
- Databases and data
- Technical illustration

OpusVL Software release pipeline

Reliable, continuous software delivery

For the DITO Project
Date: 08/06/2020
By: OpusVL Developer team, Stuart J Mackintosh
Version: v0.4



hello@opusvl.com
01788 298 450
opusvl.com

Drury House
Drury Lane
Rugby CV21 3DE

Opus Vision Limited t/a OpusVL
Company No. 03905104
VAT No. GB 747 8294 82

Open source **business management software.**
Customised to **fit the way you work.**

Contents

1 Summary

A software release pipeline enables code to travel from a developer to live services whilst adhering to the necessary governance, safety and operational requirements to deliver a reliable continuous integration and development system.

The release process enables informal releases to be made iteratively for peer review without the overhead of formalities of planned maintenance or risk of service disruption.

It is optimised for Open Source code to be imported and developed and is compatible with the Custodian model for managing non-technical aspects of software governance.

This allows regular code releases to be presented as a preview of the changes. Only when considered stable do the changes go to User Acceptance Testing and on to Live operation when appropriate.

Automated (performance, capacity, functionality) and low-friction human (fit-for-purpose) testing is supported throughout the pipeline as appropriate, depending on application and governance requirements.

Minor intricate and fine-grained code changes also use the same software pipeline ensuring the safety and quality is maintained throughout.

A software release pipeline overview is included within this document alongside a more detailed pipeline systems illustration which describes the infrastructure behind the release process.

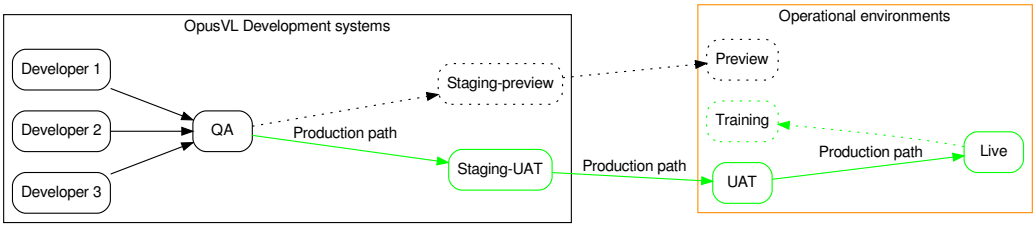
2 Software release pipeline overview

This diagram illustrates the flow of code from the developer to the various environments.

Releases travel through the production path from QA to Live and optionally to Training. This path should always be clear as any changes along this path will be swept up and deployed to live should any unplanned changes or other development project outputs require a live release.

Optional 'Preview' environments are set up to allow code to be validated outside the main release path to Live and an optional 'Training' environments. These are deployed per project and they allow experiments on an equivalent environment to Live.

This process enables the live deployment and updates to be well rehearsed therefore when releases are made to Staging and Live, the scope for failure and unplanned issues is reduced.



3 Deployment paths

Code is deployed through two paths, Production requires stringent governance and control where Preview allows for quick testing with lower overhead.

3.1 Preview

The Preview path is used to demonstrate new functionality and enable it's functional testing.

3.2 Production

This path is only used to validate that the tested code can be deployed through the Production path and is performant, not for testing functionality or new features.

It is important to keep the production path clear of untested code, as the path must remain clear for operational bug fixes and new software deployments.

Having deployments to Staging-UAT and UAT, the deployment is tested twice to prove the process prior to Live deployments. this minimises the risk of Live disruption.

4 Operational environments

To support the code release pipeline, various operational environments are required.

4.1 Developer

Purpose Enable the developer to test code as it is written
Data Dummy data / anonymised or created by the developer

- All changes and fixes applied here then promoted through the environments
- If features require disabling if not yet ready, this should be managed as a branch
- Automated testing is triggered following code commit
- Branches can only be merged when tests pass

4.2 Continuous Integration (CI)

Purpose Automated functions

- Execute unit tests
- Validate governance assets such as licence, policy
- Security scan

4.3 Quality Assurance (QA)

Purpose Post-developer testing and validation
Data Developer data / anonymised Live data

- Internal-only testing
- Validate that the release is suitable for onward deployment

4.4 Preview

Purpose To demonstrate work during a project prior to Staging release
Data Anonymised / test data

- Commissioned for each development project

- Removed at the end of the project
- Customer sign's off before released to staging

4.5 Staging

Purpose To validate solution before deploying to Live
Data Identical backport of UAT data

- Uses data representative of Live
- Staging is critical-path so should only be used when release is ready to be promoted to Live

4.6 User Acceptance Test (UAT)

Purpose Customer validation of functional readiness on customer infrastructure prior to Live deployment
Data Backport / replication of Live data / anonymised

4.7 Production-Validation

Purpose Validation for live operations
Data Live data copy

- Pre-live validation
- Proves non-functional characteristics
  - Performance
  - Deployability
  - Database compatibility
  - Security
- Operates as near-live as is feasible
  - ie separate container on live environment

4.8 Production-Live

Purpose Live operations
Data Live data

4.9 Training

Purpose Enable customer to experiment and learn about their systems without affecting Live
Data Backport / replication of Live data or anonymous / test data

5 Databases and data

Production environments will be running live data and is likely to be legally restricted through confidentiality and privacy laws.

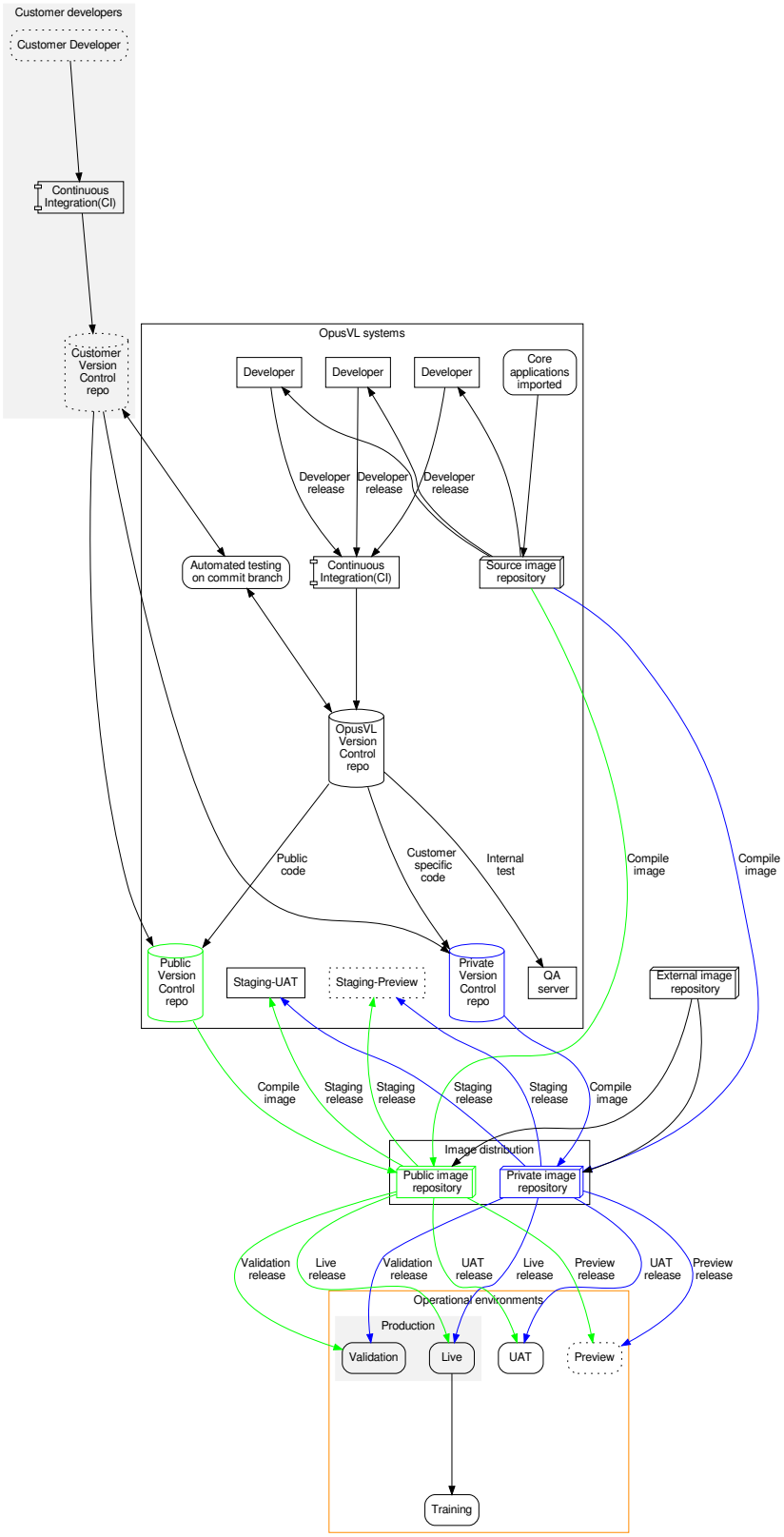
User Acceptance testing environments require data the same as, or very similar (ie anonymised Live) to production data for meaningful testing.

Training systems may have Production data or sample data depending on the training programme.

Development systems operate on disposable fabricated data and Preview should have data that enables reasonable and meaningful testing.

6 Technical pipeline illustration

The following illustration presents a detailed logical release pipeline used to transport code from the developer through to the customer production environments.


Customer developers
Customer Developer
Continuous Integration(CI)
Customer Version Control repo
OpusVL systems
Developer
Developer
Developer
Core applications imported
Developer release
Developer release
Developer release
Automated testing on commit branch
Continuous Integration(CI)
Source image repository
OpusVL Version Control repo
Public code
Customer specific code
Internal test
Compile image
Compile image
Public Version Control repo
Staging-UAT
Staging-Preview
Private Version Control repo
QA server
External image repository
Compile image
Staging release
Staging release
Staging release
Staging release
Compile image
Image distribution
Public image repository
Private image repository
Validation release
Live release
Validation release
UAT release
Live release
Preview release
UAT release
Preview release
Operational environments
Production
Validation
Live
UAT
Preview
Training

OPUSVL

Since 1999, OpusVL has helped organisations get the business management software they were looking for (but just couldn't seem to find). We combine our 'off-the-shelf' products with the craftsmanship it takes to tailor your software to your needs.

With over 20 years' experience, we're a multi-disciplined team of developers, people-people and project management experts. But, above all else, we're proud to be a team that our clients trust to get the software – and results – that they need.

opusvl.com
01788 298 450
hello@opusvl.com

Drury House
Drury Lane
Rugby CV21 3DE

Cyber Essentials

Crown Commercial Service Supplier

odoo Community Member

UKRI Innovate UK