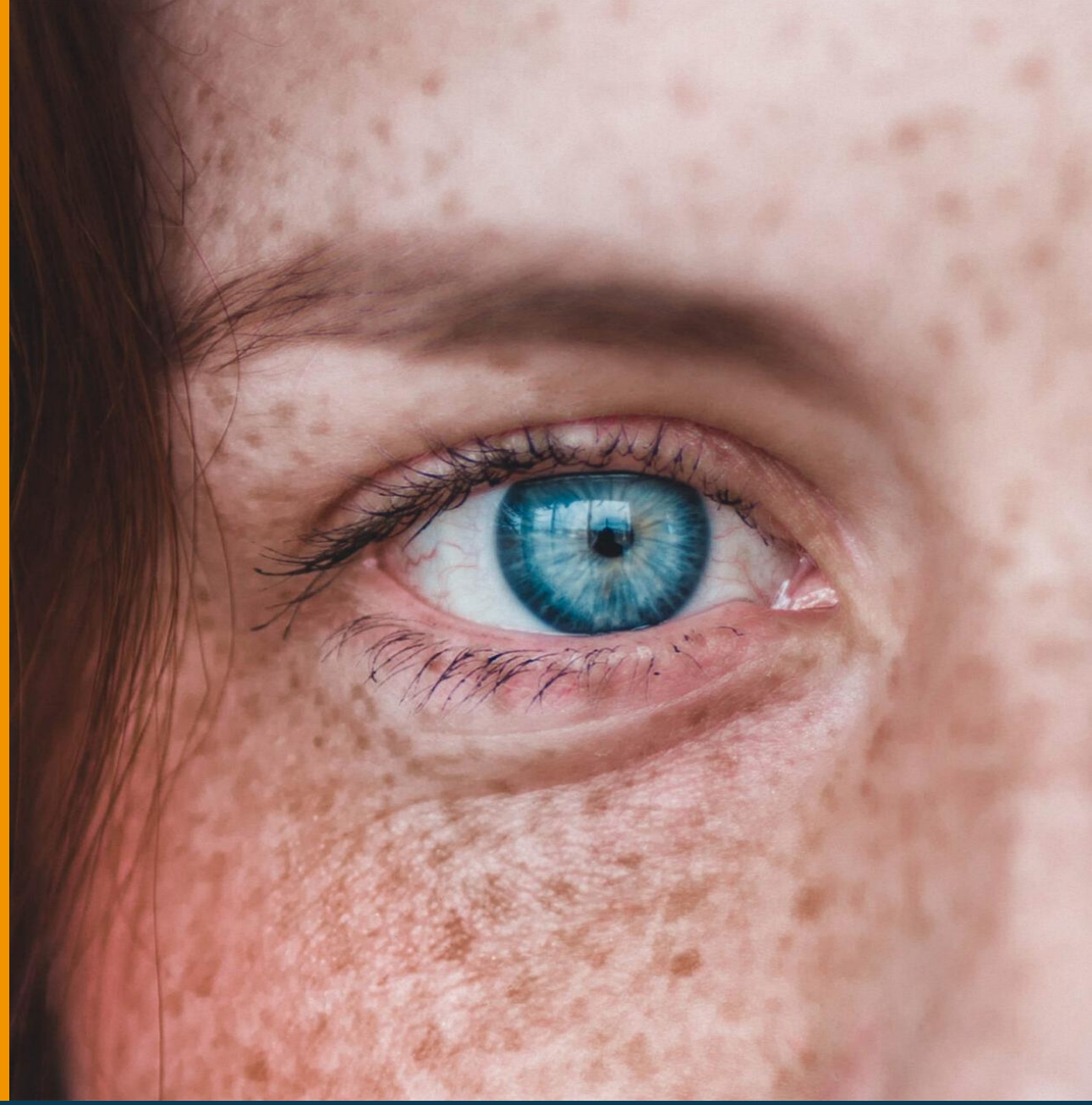
How nexeye improved its way of working with SQLcl project

SOUG DAY 2026



TRANSFER SOLUTIONS

Who am I?

Boyd Timmerman

Oracle APEX Consultant

Transfer Solutions

The Netherlands


nexeye

nexeye

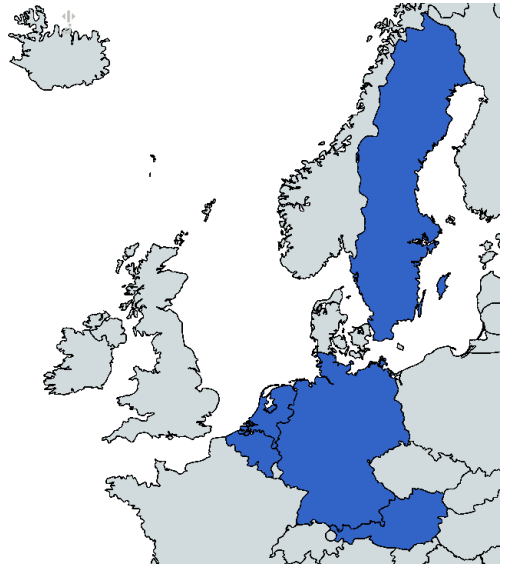
Aspires to become the market leader in the value-for-money optics segment in the European market

HANS ANDERS

DO DIREKT OPTIK

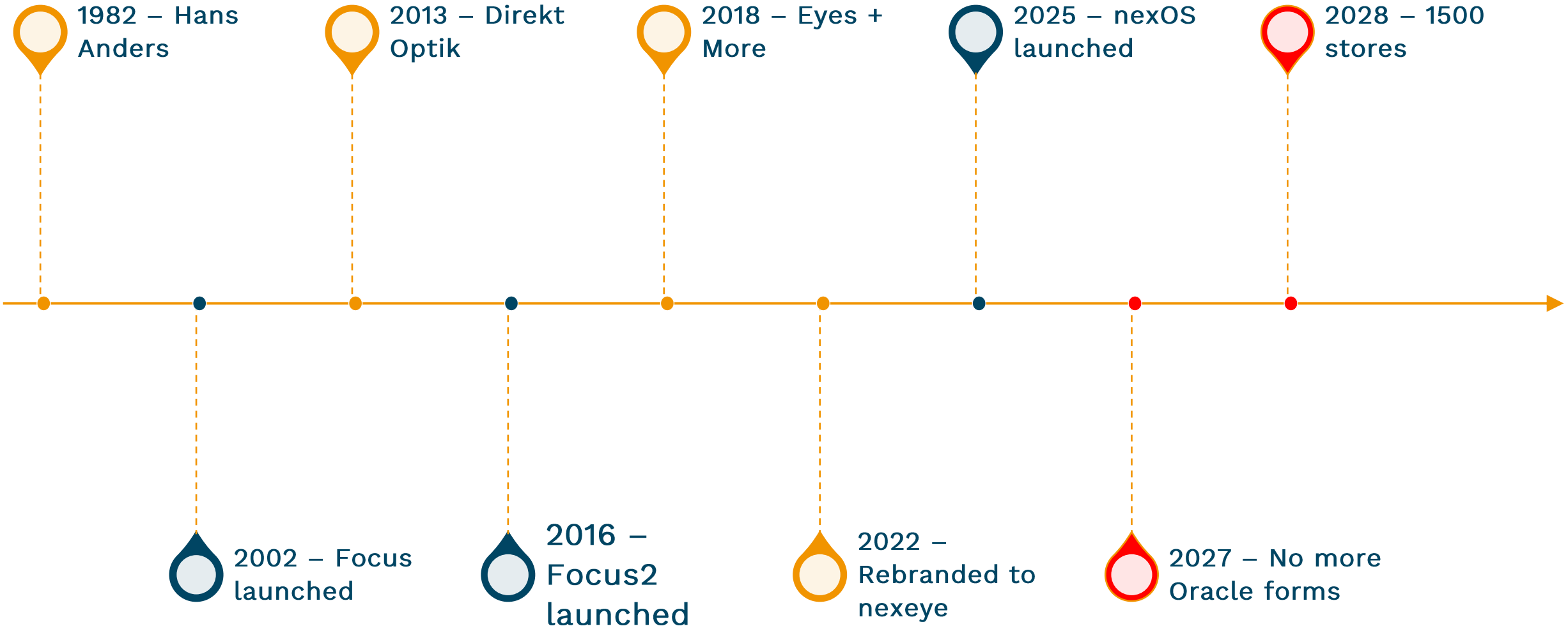
eyes more®

730 stores



5 countries

3500+
employees

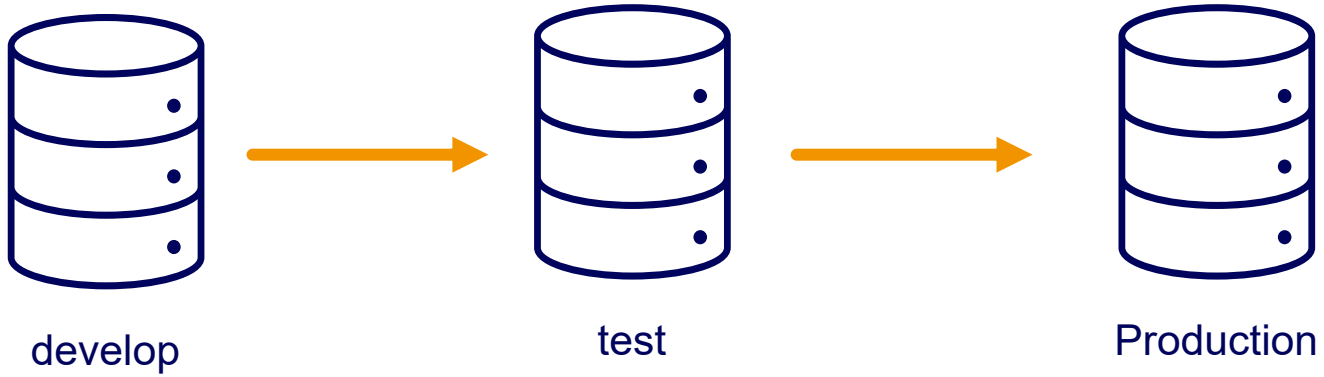

1982 – Hans Anders
2002 – Focus launched
2013 – Direkt Optik
2016 – Focus2 launched
2018 – Eyes + More
2022 – Rebranded to nexeye
2025 – nexOS launched
2027 – No more Oracle forms
2028 – 1500 stores

Situation before 2025

Situation before 2025

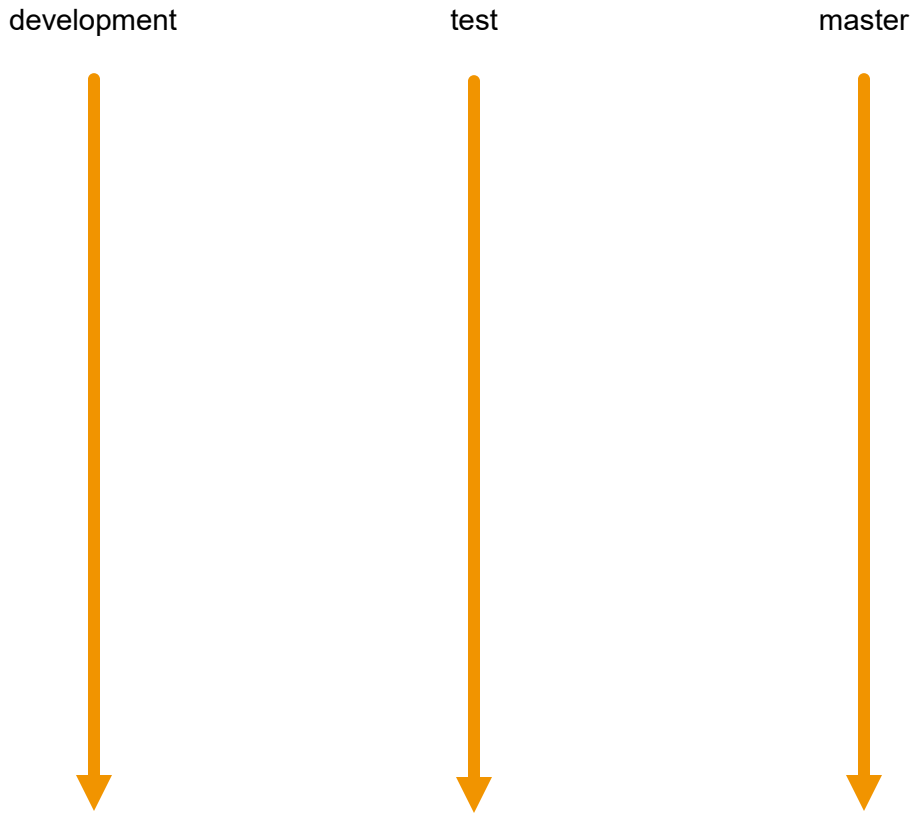
- **1 development team**
  - 3 developers
  - 1 operations manager
- **Maintenance on forms & APEX applications**
- **Changes are requested and then build**
- **Production releases at random intervals**

Environments



Version control

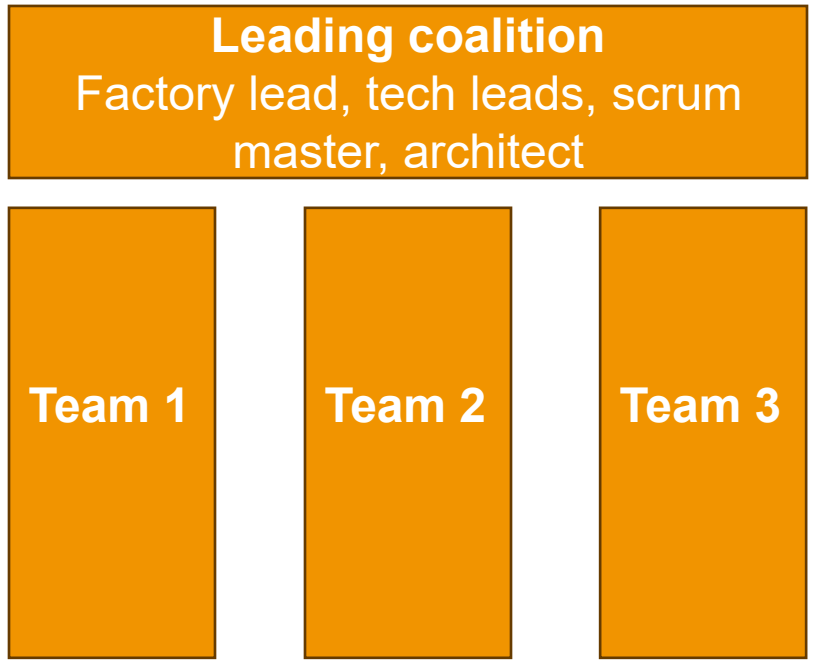
1. Back-end changes are put in feature branch
2. Application changes are put in application branch
3. Branch is merged with develop for review
4. Branch is merged with test for testing
5. Branch is merged with master for production release



Start of development factory

Start of development factory

- **4 teams**
  - 29 members
- **Scrum/agile approach**
- **2-week sprints**
- **Scheduled production release cycle**



Key bottlenecks

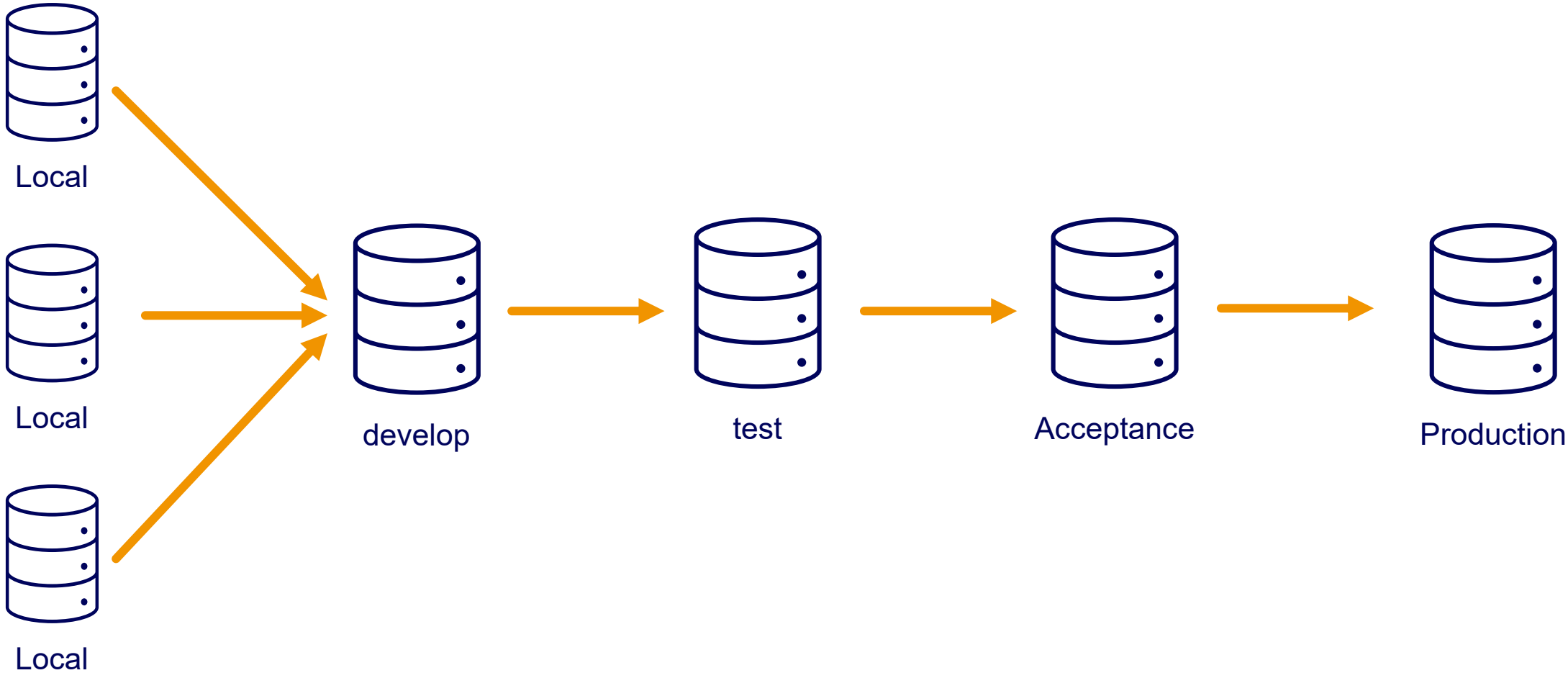
- Shared development database environment results in conflicting changes and work invalidation
- APEX application modifications and exports risk introducing unintended features across environments
- Absence of acceptance testing environment prevents production deployment validation
- Production release cycle requires approximately 60 hours of effort

Solutions and improvements

1. Environment changes
2. Version control
3. SQLcl project
4. Release pipelines

Environment changes

Current environments

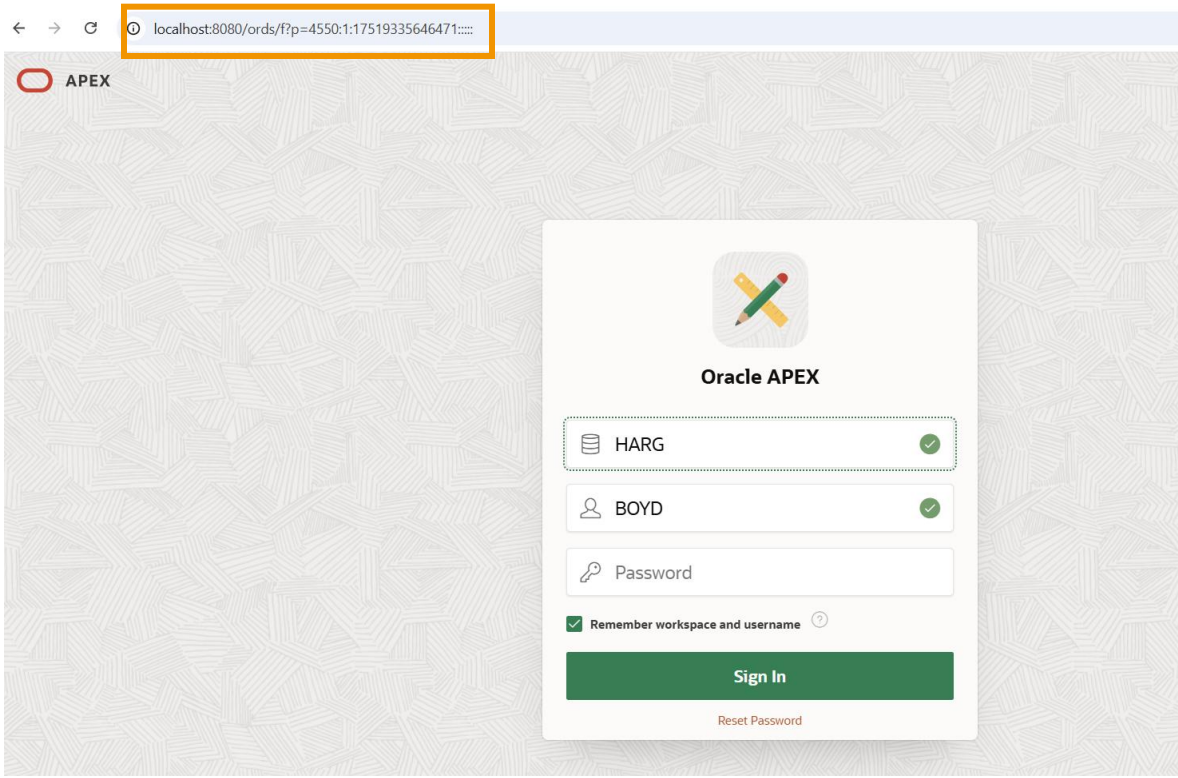


Local development

- Isolated development environment
- Simplified version control
- Support for parallel development streams

Local development

- **Every developer has installed on its device**
  - Oracle 19c EE; patch 19.26
  - Oracle APEX 21.2
  - Tomcat 9
  - ORDS 21.4
- **Connects to localhost**

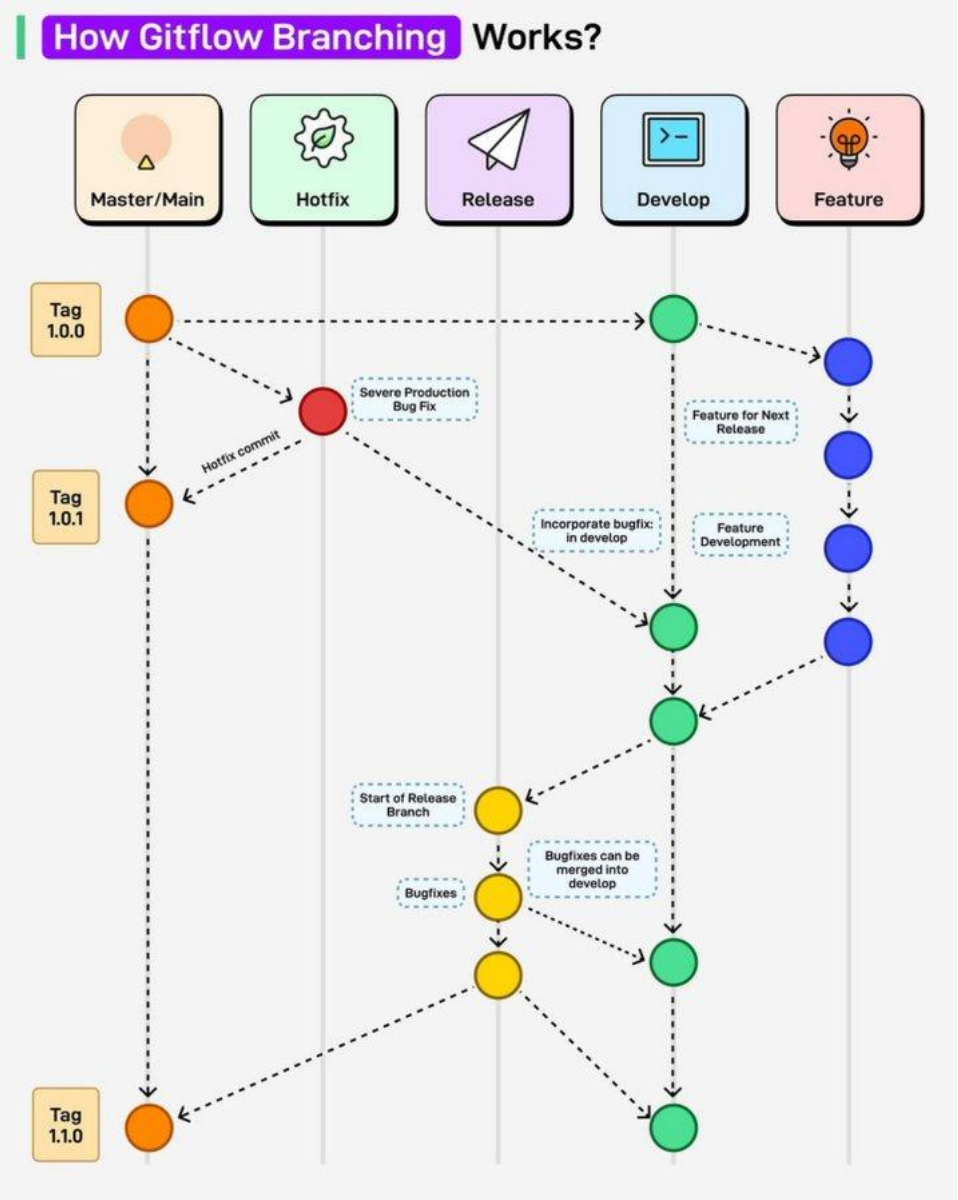


Version control

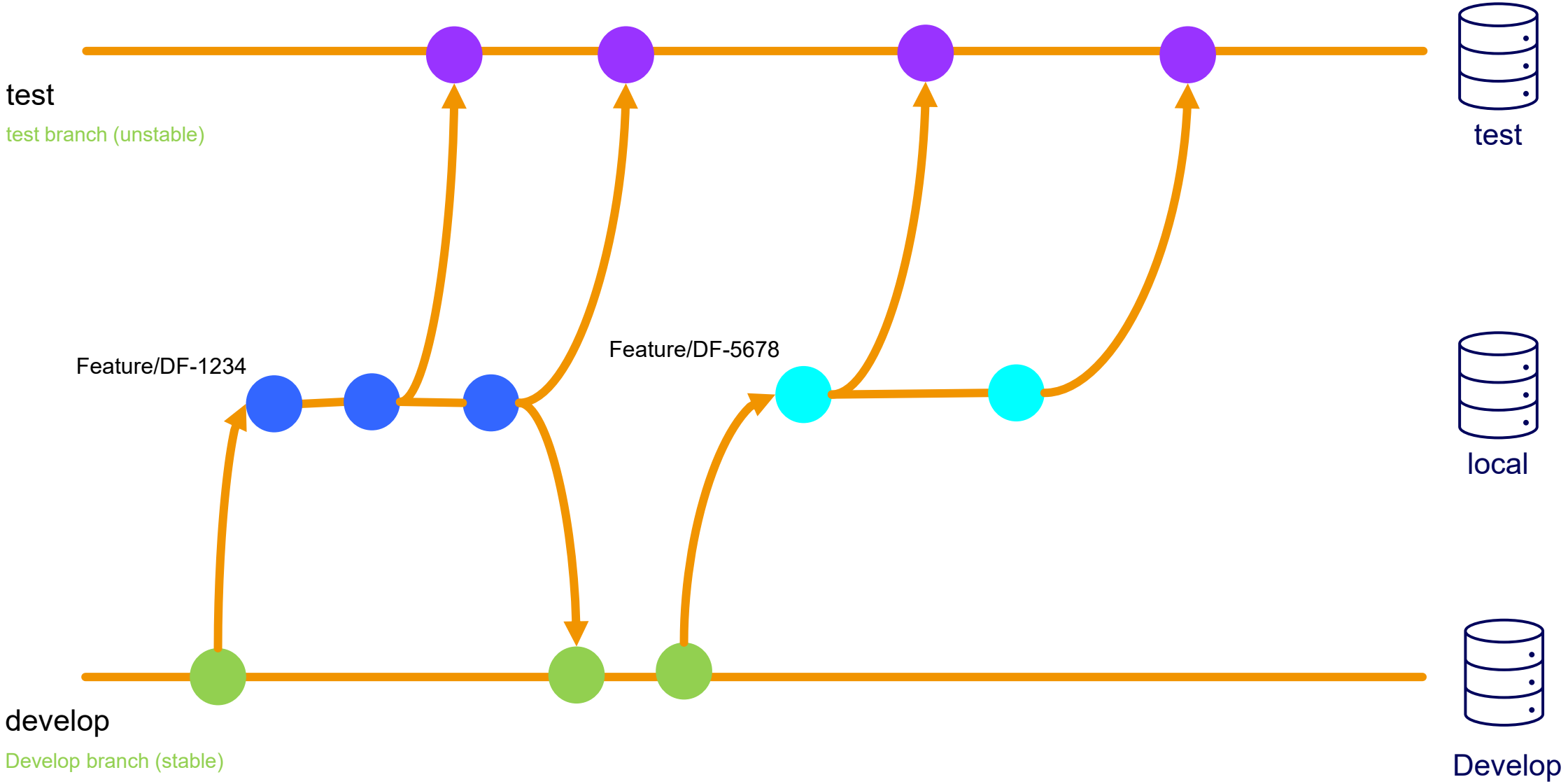
New branching strategy

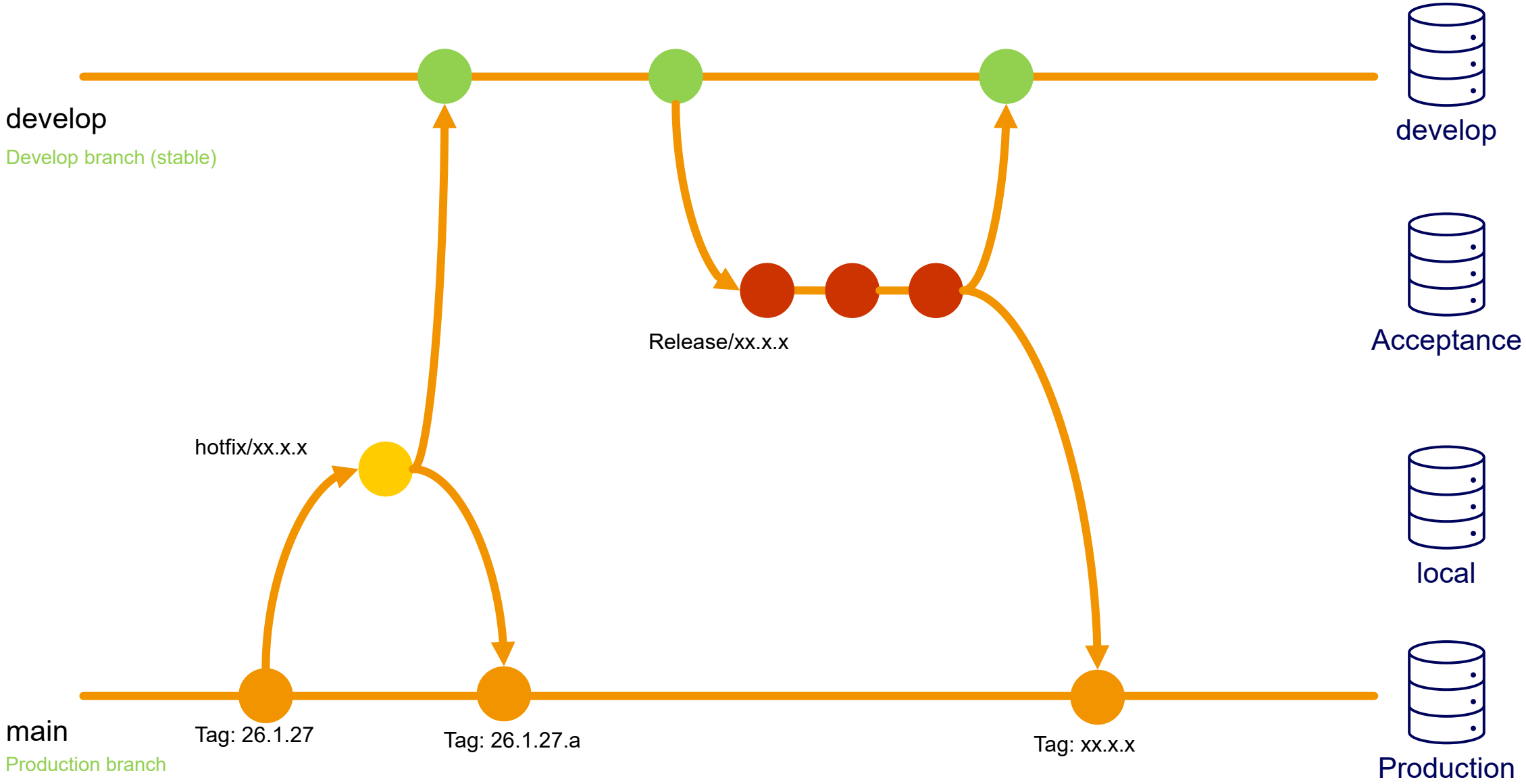
Gitflow

1. Development starts on feature branch
2. Features are merged into develop after functional testing
3. Release branch is created from develop
4. Release branch is merged into main
5. Urgent fixes are created from main



https://x.com/Python_Dv/status/1967817975710257644


test
test branch (unstable)
test
Feature/DF-1234
Feature/DF-5678
local
develop
Develop branch (stable)
Develop


develop
Develop branch (stable)
develop
Release/xx.x.x
Acceptance
hotfix/xx.x.x
local
main
Production branch
Tag: 26.1.27
Tag: 26.1.27.a
Tag: xx.x.x
Production

Gitflow

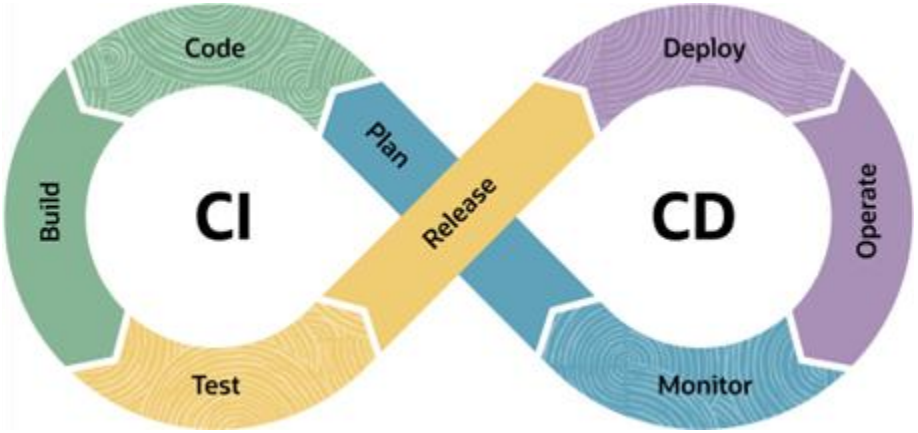
- Centralized Feature Development
- Accelerated Code Availability
- Pre-Production Quality Assurance

SQLcl project

SQLcl project

- The CI/CD tool from Oracle
- Based on liquibase

- Versioned feature management
- Release artifact creation
- Source control integration
- Automated release packaging
- Ordered installation

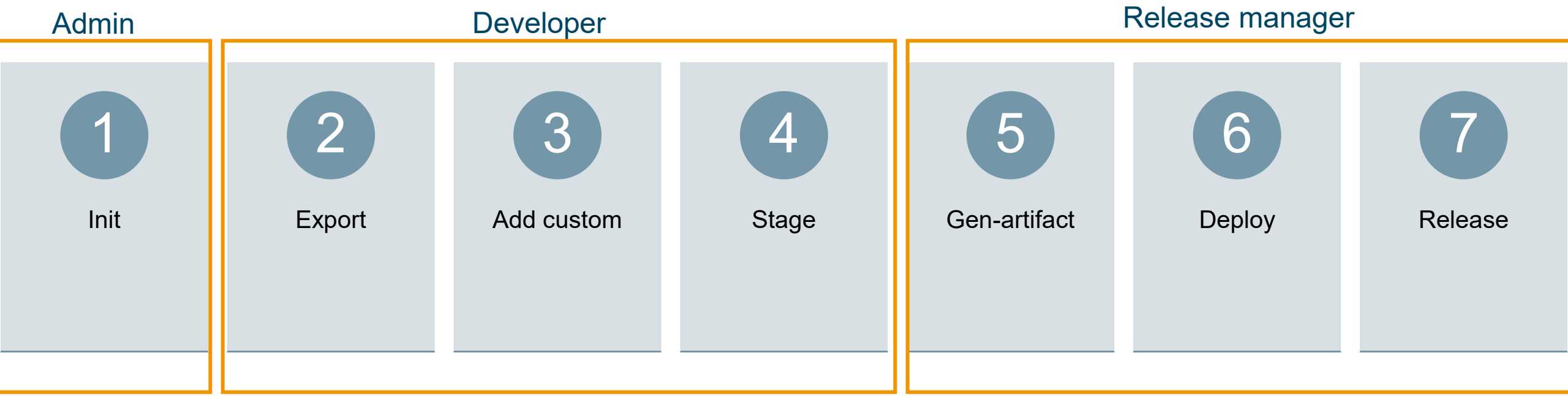


https://docs.oracle.com/en/database/oracle/sql-developer-command-line/24.4/

Project components

1. Init
2. Export
3. Add custom
4. Stage
5. Gen-artifact
6. Deploy
7. Release

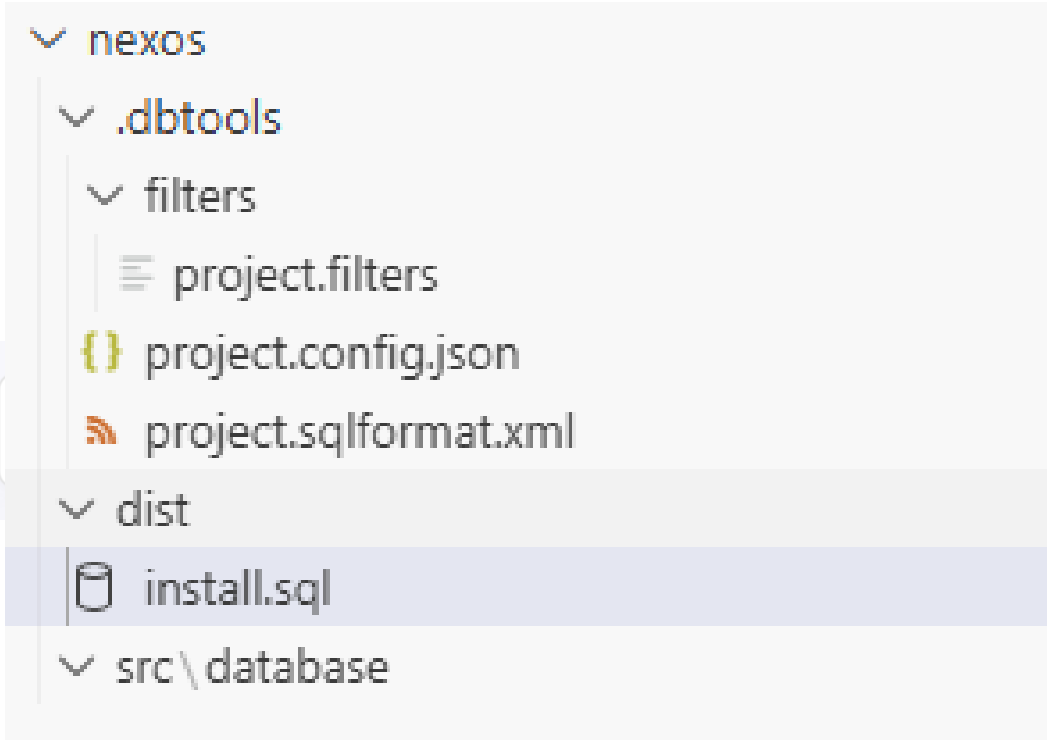
Project components



Admin steps

1. Init

- **Creates folder structure**
  - *src* & *dist* are important
- **Creates config files**
  - project.filters
  - project.config.json
  - project.sqlformat.xml



Config files

- project.config.json

```
{
  "project" : "DEMO",
  "sqlcl" : {
    "connectionName" : "",
    "version" : "25.2.2.0"
  },
  "schemas" : [ "PROJECT" ],
  "lastReleaseVersion" : "1.0",
  "export" : {
    "format" : {
      "enable" : true
    },
    "setTransform" : {
      "collationClause" : "NEVER",
      "constraints" : true,
      "constraintsAsAlter" : true,
      "emitSchema" : false,
      "force" : true,
      "partitioning" : false,
      "segmentAttributes" : false,
      "sizeByteKeyword" : true,
      "sqlterminator" : true,
      "storage" : false,
      "tablespace" : false
    }
```

Config files

project.filters

```
-- Liquibase Tables
object_type != 'TABLE' or object_name not in ('DATABASECHANGELOG',
                                               'DATABASECHANGELOGLOCK',
                                               'DATABASECHANGELOG_ACTIONS'
                                               ),
not (object_type = 'VIEW'    and object_name ='DATABASECHANGELOG_DETAILS'),
not (object_type = 'TRIGGER' and object_name ='DATABASECHANGELOG_ACTIONS_TRG'),


-- DM generated tables
--not (object_type = 'TABLE' and object_name like 'DM$%' ),
--not (object_type = 'VIEW' and object_name like 'DM$V%' ),
object_name not like 'DM$%',      -- covers tables, views, indexes
object_name not like 'I_MLOG$%',      -- covers Materialized Views Logs Indexes
object_name not like 'I_SNAP$%',

export_type not in ('USER'),
```

Development steps

Development steps

1. Create feature branch
2. Refresh local DB
3. Develop feature
4. Export changes
5. Stage changes
6. Create pull request

1. Create feature branch

- **Branch naming convention**
  - feature/
  - hotfix/
  - bugfix/

```
1  git checkout -B feature/DF-1234
```

2. Refresh local database

- Run build.sh script
- Imports datapump from production
- Deploys changes from git

```
BoydTimmerman@NLNB1225 MINGW64 /c/nexos (featureDF-3898)
$ ./build.sh
```

```
[SUCCESS] 2026-02-18 14:47:20 - Step 6: Artifact deployment completed successfully
[INFO] 2026-02-18 14:47:20 - Cleaned up artifact file
[SUCCESS] 2026-02-18 14:47:21 - ========================================
[SUCCESS] 2026-02-18 14:47:21 - Build Script Execution Completed Successfully!
[SUCCESS] 2026-02-18 14:47:21 - ========================================
[INFO] 2026-02-18 14:47:21 - Total execution time: 00:03:28 (HH:MM:SS)
[INFO] 2026-02-18 14:47:21 - Log file saved to: /c/nexos/logs/build_20260218_144353.log
[SUCCESS] 2026-02-18 14:47:21 - ========================================
```

build.sh

1. Kill sessions & drop users
2. Import .dmp files to recreate production
3. Reinstall APEX application if needed
4. Deploy SQLcl project artifact
5. Check for invalid objects and recompile

```
[SUCCESS] 2026-02-18 14:47:20 - Step 6: Artifact deployment completed successfully
[INFO] 2026-02-18 14:47:20 - Cleaned up artifact file
[SUCCESS] 2026-02-18 14:47:21 - ========================================
[SUCCESS] 2026-02-18 14:47:21 - Build Script Execution Completed Successfully!
[SUCCESS] 2026-02-18 14:47:21 - ========================================
[INFO] 2026-02-18 14:47:21 - Total execution time: 00:03:28 (HH:MM:SS)
[INFO] 2026-02-18 14:47:21 - Log file saved to: /c/nexos/logs/build_20260218_144353.log
[SUCCESS] 2026-02-18 14:47:21 - ========================================
```

3. Develop feature

- Developers compiles changes on database
- Developers make changes in APEX applications

4. Export

- When development is done, it’s time to export
- All changes are put into *src*-folder
- Add –o for specific objects

4. Export

- **Custom scripts**
  - DML
  - disables

```
project stage add-custom -file DF-1234_ins_users
```

5. Stage

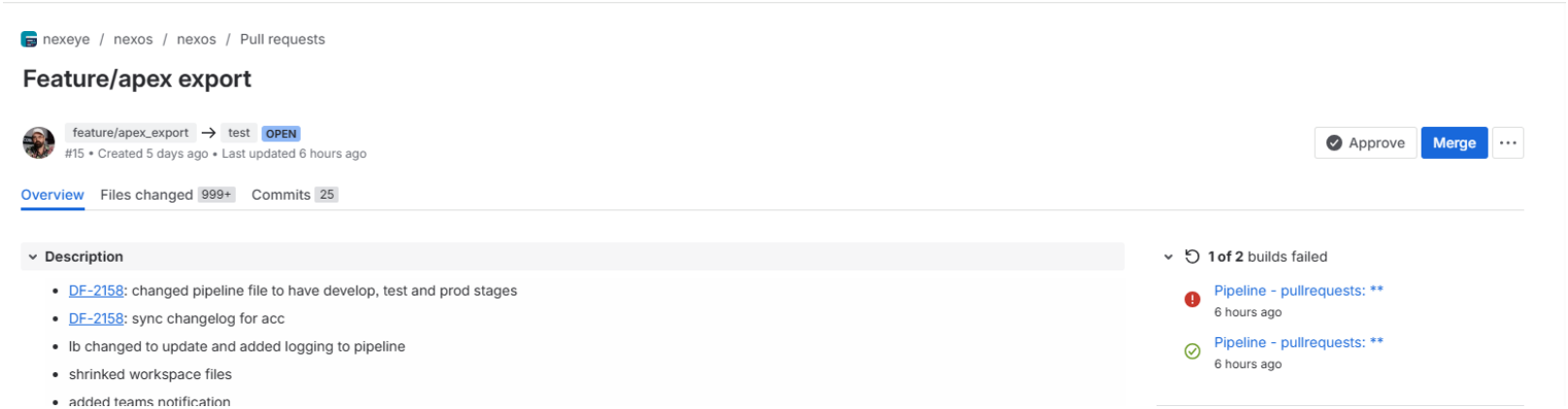
- **Stage changes for deployment**
- **Differences between feature branch and develop branch are put in *dist*-folder**

```
1  project stage -verbose
2  !git add --all
3  !git commit -m "DF-3898: staged changes"
4  !git push
```

- **Review the files**
  - Sometimes the ordering can be off

6. Pull request

- A pull request into develop is created
- Peer review can be performed
- After merging the code can be deployed



Release steps

Release pipelines

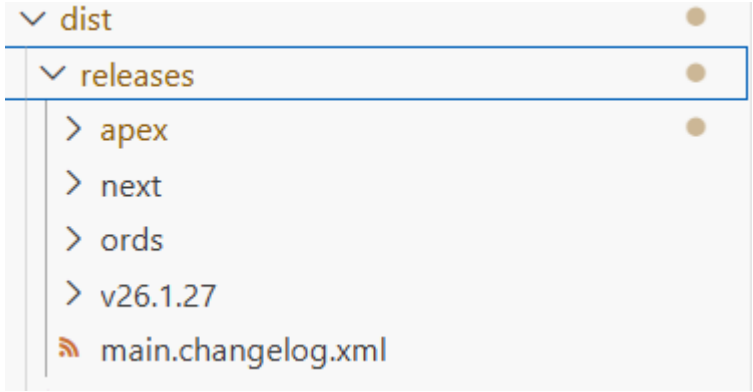
Release pipelines

- Bitbucket pipelines
- Each branch has its own pipeline
- Using *project* commands to deploy code

```
pipelines:
    branches:
        test:
          - step:
              runs-on:
                - self.hosted
                - linux
                - 'eks'
              name: Deploy Database Artifact
              deployment: test
              script:
                - echo "Deploying test branch to Test Environment"
                - echo "Starting artifact generation and deployment..."
                - |
                  sql ${TS_DB_USERNAME}/${TS_DB_PASSWORD}@${TST_DB_CONNECTION} <<EOF
                  set define off
                  project gen-artifact -version deploy
                  project deploy -file artifact/nexos-deploy.zip -verbose
                  exit;
                  EOF
                - echo "Artifact generation and deployment completed."
```

Releasing

- Create a release, by using the *release* parameter
- All features are bundled in the release folder
- The *next*-folder is cleared




```
1  sql -name local
2
3  project release -version 26.1 -v
```

Added value

Improvements

- Developers can now work in parallel without interfering with one another
- Release preparation time has been reduced from approximately 60 hours to 1 hour
- Production deployments are now validated and predictable, rather than performed without prior verification
- Release content is fully transparent, with clear visibility into exactly what is being deployed
- Feature delivery throughput has increased significantly

Challenges

- Developers need to fully understand the way of working
- If the *develop*-branch contains invalid every build will fail
- Even if you’re developing locally the branches should always be up-to-date
- Merging APEX continues to be challenging

Questions?

More info here

btimmerman.hashnode.dev

github.com/boydtimmerman

linkedin.com/in/boyd-timmerman