Releasemanagement mit gCTS

rku IT

rku-it.de

Wir auf einen Blick

Vorstellung der rku.it GmbH

seit **1961** in Herne	> **130** Kunden	**~ 92,1 Mio. €** Umsatz	**> 80** Produktivsysteme mit ERP-Anwendungssoftware
16 Gesellschafter	**Utilities, Transport, Public**	**~ 5 Mio.** Zählpunkte	**> 6.500** administrierte Endgeräte
~ 450 Mitarbeitende	**deutschland-weit**	**> 850** Windows-Server	**> 15.000** Nutzende von Anwendungs-software

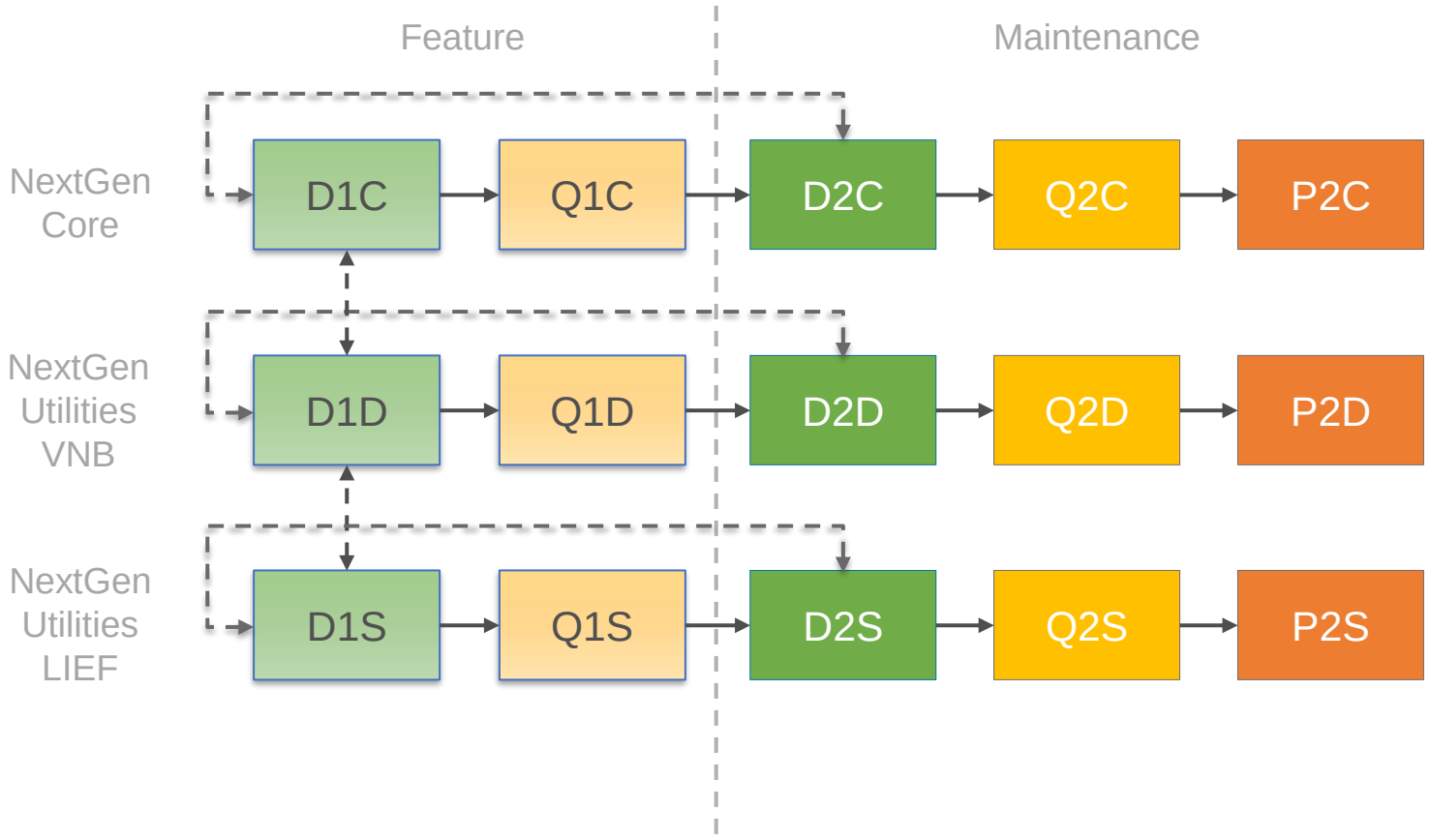
Releasemanagement mit gCTS

Inhaltsverzeichnis

Releasemanagement mit gCTS

Systemlandschaft

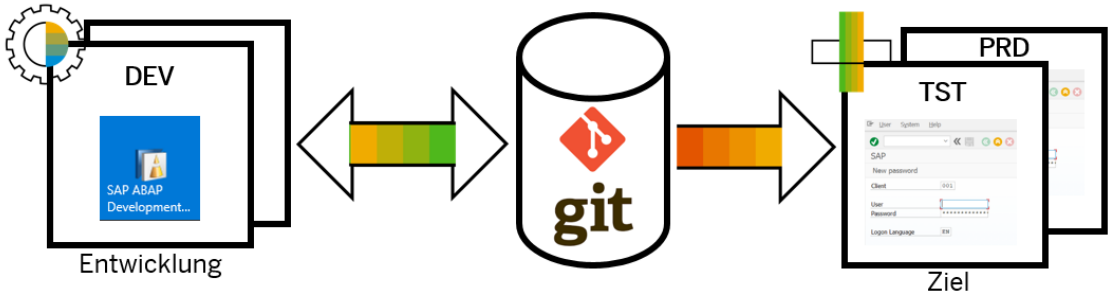
- On-Premise Systemlandschaft mit drei wesentlichen Systemschienen
- Alle Systeme sind Mehrmandantensysteme
- Customizing teilweise harmonisiert, d.h. zentral von einem Mandanten verteilt

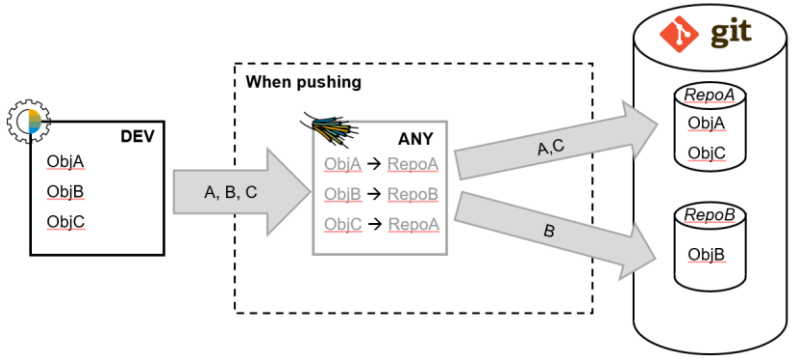


Releasemanagement mit gCTS

Basics: Was ist gCTS?

- Git-enabled Change and Transport System
- gCTS seit S/4HANA 1909
- Zentrale Registry seit S/4HANA 2021
- Entwicklung und Customizing
- Integration in SAP-Standardtransportprozess
- Versionsverwaltung in Git, Synchronisation mit SAP-Systemen über Transportaufträge/Transport von Kopien





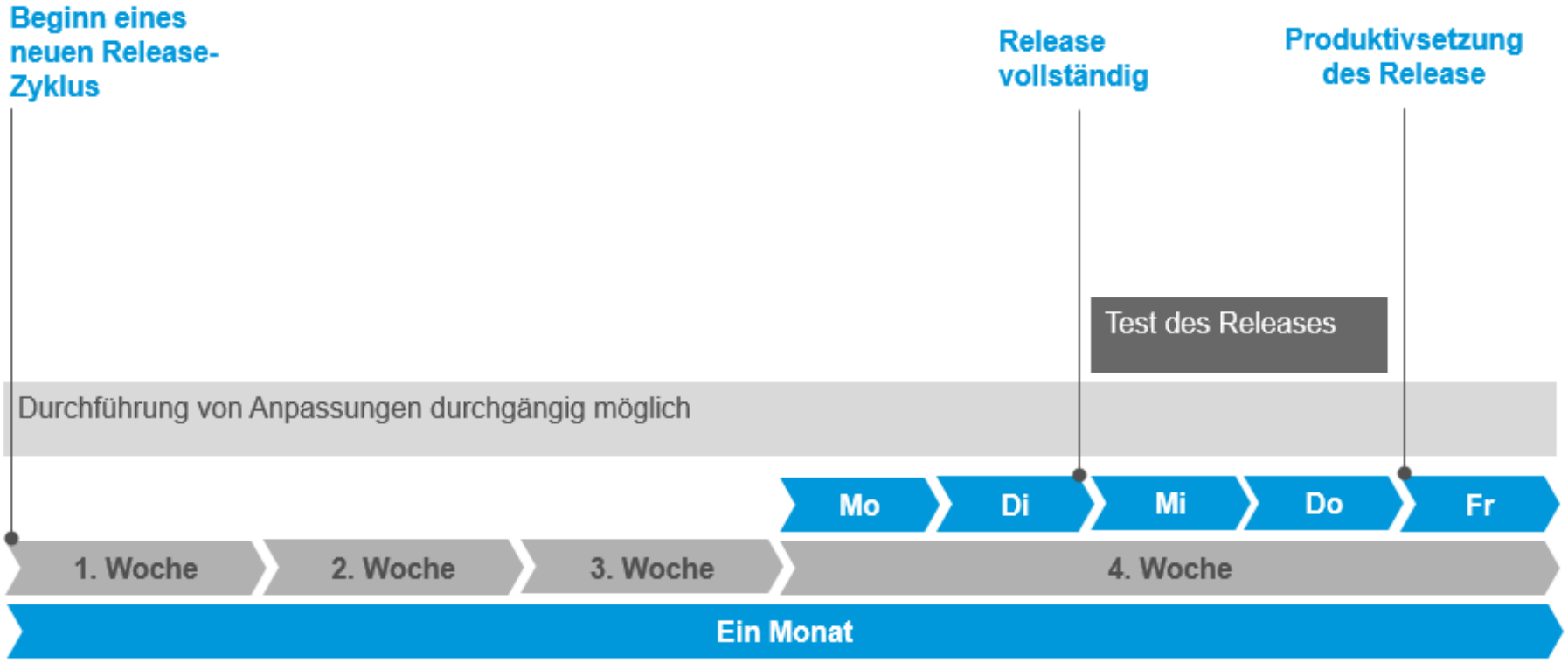
Releasemanagement mit gCTS

Definition: Was verstehen wir unter Releasemanagement?

- Regelmäßige Produktivsetzung
  - Major Release (zwei- bis viermal im Jahr, z.B. neues S/4HANA Release, Formatwechsel)
  - Minor Release (einmal im Monat)
  - Hotfixes (nach Bedarf, Produktivsetzung jederzeit möglich)
- Planung und Historie
  - Inhalte eines Releases werden in Abstimmung mit dem Anforderungsmanagement und den Product bzw. Service Ownern geplant
  - Für jedes Release werden Release Notes veröffentlicht. Dadurch ist auch die Historie der Weiterentwicklungen nachvollziehbar
- Risikomanagement
  - Bewertung der Abhängigkeiten und Auswirkungen, Einsatz geeigneter Kontroll- und Eindämmungsmaßnahmen

Releasemanagement mit gCTS

Releasezyklus eines Minor Releases



rku IT

Inhaltsverzeichnis

Releasemanagement mit gCTS

Rahmenbedingungen I: Testmanagement

- Enge Interaktion mit dem Testmanagement
  - Vor Major Releases müssen End-To-End-Tests gemäß Testkatalog als Qualitity Gate durchgeführt werden. Dafür ist ein Zeitraum von ein bis zwei Wochen zu planen.
  - Der nötige Testumfang für Minor Releases ist anhand der Inhalte zu bewerten.
    In der Regel werden isolierte Anwendungs-, Prozess- oder Integrationstests ausreichen.
  - Voraussetzung für den reibungslosen Ablauf der Tests und eine niedrige Fehlerquote, sowohl während der Entwicklungsphase als auch zum Release, ist das Vorhandensein von Testdaten in guter Qualität und Aktualität auf beiden Testsystemen.

Rahmenbedingungen II: Qualitätssicherung

- Absicherung der Releasefähigkeit durch adäquate Qualitätssicherheitsmaßnahmen
  - Automatisierte Überprüfung des Custom Codes durch statische Prüfungen (ATC Checks)
  - Automatisierte Überprüfung des Custom Codes durch dynamische Prüfungen (Unit Tests, Regressionstests)
  - Manuelle Überprüfung des Custom Codes, z.B. durch Code Review

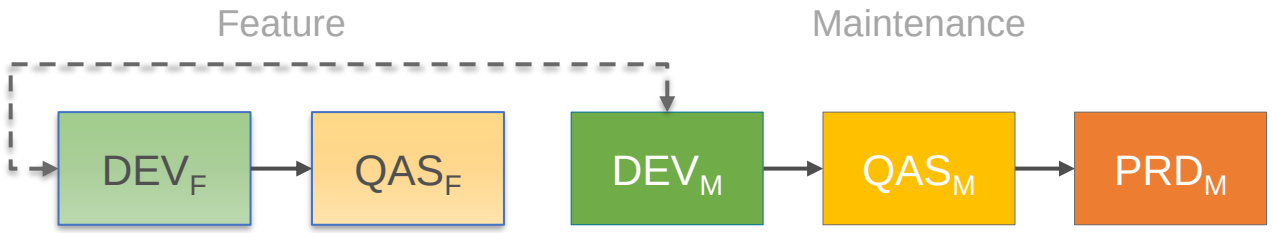
- Bewertung der Risiken durch den Releasemanager
  - Übersicht über den Scope des Releases
  - Impact Analysis
  - Abhängigkeitsanalyse

Releasemanagement mit gCTS

Inhaltsverzeichnis

Releasemanagement mit gCTS

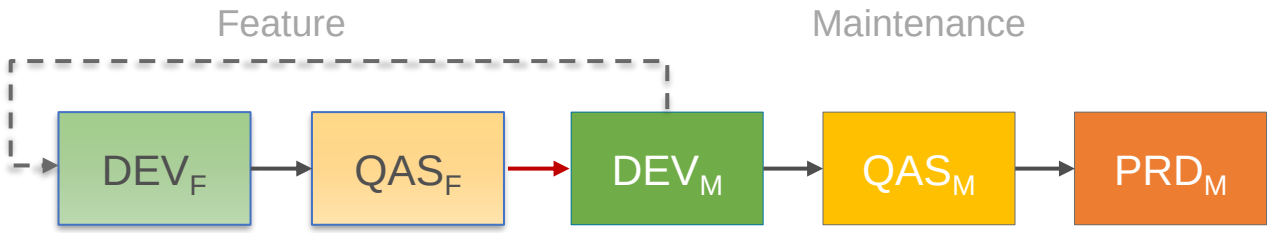
Szenario 1: Einzelne Systemschiene (Feature und Maintenance)



- Alle Entwicklungen für Major und Minor Releases finden auf DEV$_F$ statt, die Tests auf QAS$_F$. Die Transporte erfolgen mittels klassischem CTS oder Transportmanagementtool.
- Zum Abschluss des Releases findet je Repository ein Merge aller abgenommen Änderungen in den Main Branch statt. Danach Übertragung nach DEV$_M$ mittels gCTS.
- Je nach Größe und Kritikalität des Releases finden übergreifende Abnahmetests auf QAS$_M$ als Quality Gate statt. Ansonsten wird das Release unverzüglich nach PRD$_M$ weiter transportiert, um einen möglichst einheitlichen Stand in der Maintenance-Schiene zu gewährleisten.
- Entwicklungen für Hotfixes finden auf DEV$_M$ statt und werden auf QAS$_M$ getestet. Nach Freigabe werden sie unverzüglich produktiv gesetzt und die Änderung in den Main Branch übernommen.

Releasemanagement mit gCTS

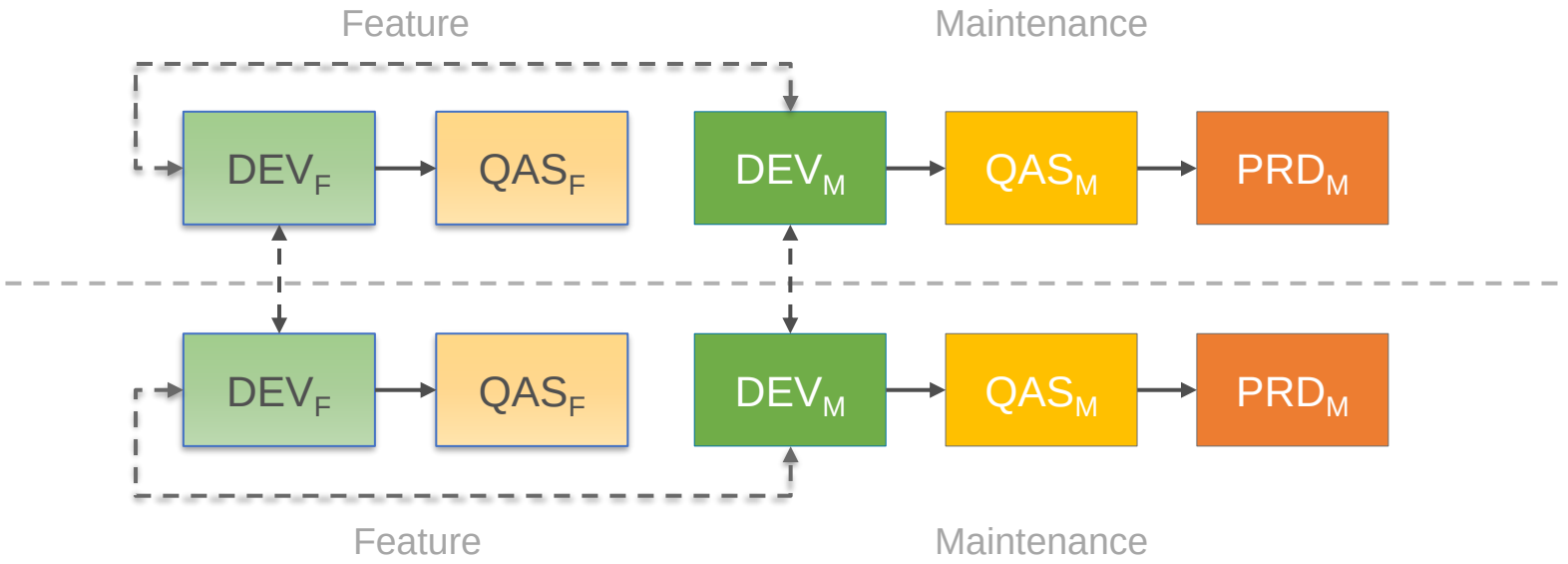
Szenario 1a: Einzelne Systemschiene (Feature und Maintenance)



- Die Systemrollen und regulären Transportwege bleiben identisch.
- Die Übertragung des Releases von der Feature- in die Maintenance-Schiene erfolgt nicht mittels gCTS, sondern mit den Mitteln des Transportmanagementtools.
- Der Retrofit der Hotfixes von DEV$_M$ nach DEV$_F$ erfolgt weiterhin über gCTS, oder für einzelne Objekte direkt mittels der Vergleichsfunktion von ADT.

Releasemanagement mit gCTS

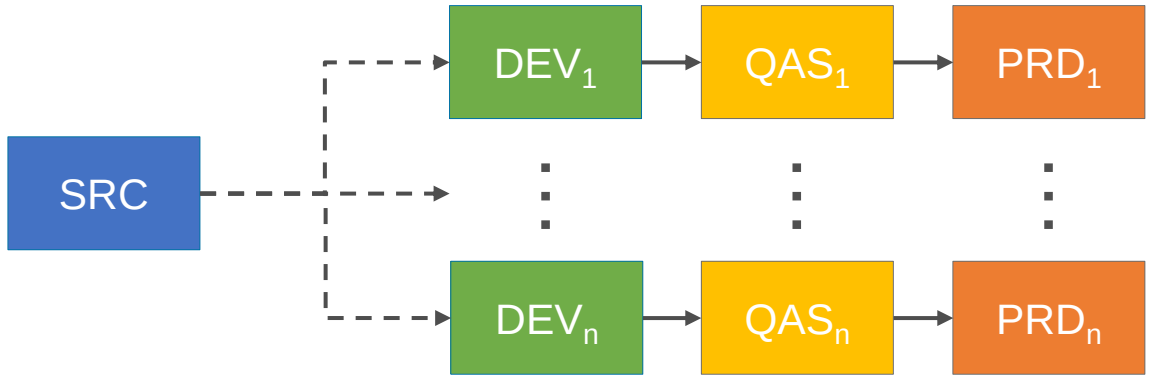
Szenario 2: Mehrere parallele Systemschienen



- Systemübergreifend genutzte Objekte werden mittels gCTS synchronisiert.
- Ein Branch wird immer nur von einem System genutzt und alle Branches werden regelmäßig mit dem Main Branch synchronisiert, um produktive Anpassungen aus anderen Systemschienen aufzunehmen.
- Nach Möglichkeit wird je Repository ein System als führendes System ausgewählt.

Releasemanagement mit gCTS

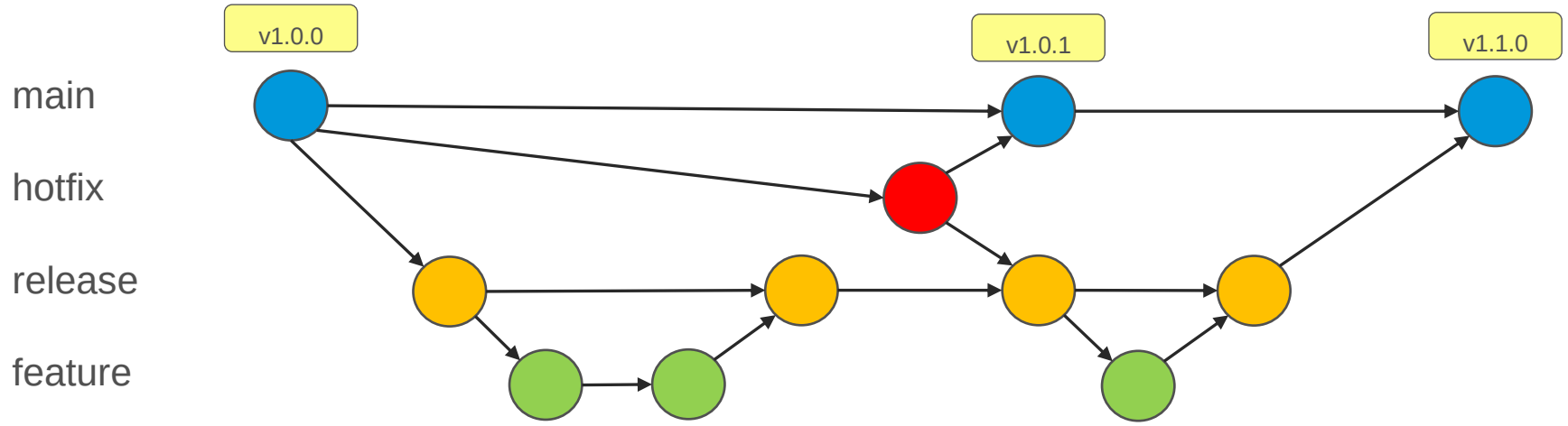
Szenario 3: Produktentwicklung (ein Quellsystem, viele Zielsysteme)



- Repositories werden in SRC in der Rolle „Entwicklung“ angelegt, in den anderen Systemen in der Rolle „Bereitgestellt“
- Die Zielsysteme DEV_1, …, DEV_n beziehen die Inhalte über Git, der weitere Transport kann individuell gestaltet werden.
- Fehlermeldungen und Änderungswünsche werden über Issues an die Entwickler mitgeteilt.
- In manchen Fällen könnte auch eine direkte Fehlerkorrektur in einem der DEV-Systeme Sinn ergeben. In diesem Fall kann die Korrektur über einen Pull Request nach SRC übernommen werden.

Releasemanagement mit gCTS

Branching Modell (Konzept)



- Main Branch enthält immer den produktiven Stand
- Branch „Hotfix“ zur Entwicklung von Fehlerkorrekturen als aktiver Branch im Maintenance-System; Merge in den Main Branch bei Produktivsetzung, dann Merge in den Release Branch
- Branch „Release“ zur Sammlung aller Anpassungen für das nächste Release; Merge in den Main Branch bei Produktivsetzung des Releases
- Branch „Feature“ für neue Entwicklungen als aktiver Branch im Feature-System; Merge in den Release Branch nach Freigabe

Releasemanagement mit gCTS

rku IT

Zuschnitt und Größe von Repositories

Vor der Einführung der zentralen Registry gab es eine 1:1-Beziehung zwischen gCTS-Repositories und Transportschichten im Entwicklungssystem.

Mit der Einführung der zentralen Registry ist dies nun granularer steuerbar:

- Softwarekomponente oder Transportschicht
- Paket(-hierarchie)
- Explizite Objektliste

Im Sinne des Releasemanagements erscheint die Verwendung eines Repositories je Softwarekomponente aber nach wie vor in den meisten Fällen optimal.

Releasemanagement mit gCTS

Inhaltsverzeichnis

Releasemanagement mit gCTS

Herausforderungen

- Parallele Entwicklung
  - Im Rahmen des Tages- und Projektgeschäfts kann es dazu kommen, dass ein Service in unterschiedlichen Systemschienen, oder Feature und Maintenance geändert wird. Diese Anpassungen müssen koordiniert und konsolidiert werden.
  - Änderungen an einem Service können aus unterschiedlichen Anforderungen oder Projekten erwachsen, die unterschiedliche Zeithorizonte haben. Diese müssen ebenfalls koordiniert und konsolidiert werden.

- Know-How-Aufbau (intern und extern!)
  - Git und das gCTS sind für viele ABAP-Entwickler Neuland. Dadurch entsteht erhöhter Aufwand bei der Schulung interner und dem Onboarding externer Mitarbeiter.

Releasemanagement mit gCTS

Inhaltsverzeichnis

Releasemanagement mit gCTS

rku IT

Demonstration

Beispiel: Lebenszyklus eines Services

Die Demo zeigt Anpassungen eines Service im Feature- und Maintenance-System während eines Releasezyklus.

Folgende Aktivitäten werden gezeigt:

- Commit durch Freigabe eines Hotfix im Maintenance-System
- Merge in den Main Branch bei Produktivsetzung
- Retrofit durch Merge in den Feature-Branch
- Commit durch Freigabe im Feature-System
- Merge in den Main Branch im Rahmen des Releases
- Merge in den Maintenance Branch im Rahmen des Releases

Weitere Informationen

Links

- SAP-Dokumentation zu gCTS:
  Git-fähiges Change and Transport System (gCTS) (BC-CTS-GIT) | SAP Help Portal

- SAP-Beispielimplementierung des BAdIs zur Nutzung der zentralen Registry:
  GitHub - SAP-samples/s4hana-gcts: Sample code to integrate gCTS in the development workflow.

Ihr Ansprechpartner

Dr. Christopher Graw

Developer

Telefon: +49 2323 3688-507
Mobil: +49 1590 4076295
eMail: christopher.graw@rku-it.de