APACK
A Package and Dependency Manager for ABAP

Sebastian Wolf, Development Architect - Central Engineering, SAP SE

@Ygriega

May 2019

THE BEST RUN **SAP**

ABAP – A Special "Snowflake"

- ABAP: Over 30 years of history
- Considerable evolution especially in recent years
- However: Special characteristics have remained
  - In-system development
  - No local deployment/development
  - Full-fledged ABAP instance required for development
  - System applies two object states: Active/Inactive
  - System-central, proprietary version control system
  - No explicit dependency management
  - Weak, but complex package management
- abapGit to the rescue?!
  - Community-driven distributed version control for ABAP
  - Adopted in SAP Cloud Platform
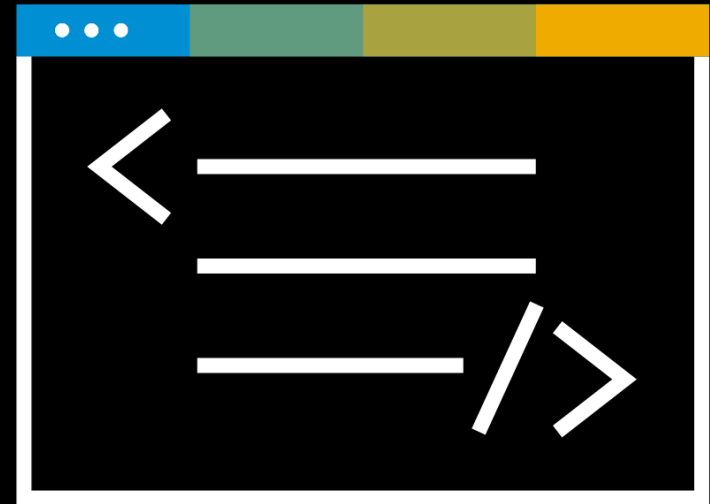
ABAP on SAP Cloud Platform (Steampunk) – State-of-the-Art

- Custom code can be imported via abapGit
- Artifact transport between instances with gCTS

Whitelist Implications

- Custom code can only use whitelisted objects
- Whitelist is still very restricted
- → Ports of existing solutions and new developments are challenging

Porting existing foundation libraries

- Re-use libraries will need to be built
- Alternative: Port/Adopt re-use functionality from not-whitelisted components
- Required libraries can currently only be referenced via README
  → **Very error-prone and cumbersome!**

Wishes and Questions by the abapGit Community

- https://github.com/larshp/abapGit/issues/217
- Discussed since May 2016 (maybe longer…)

Requirements

- Deploy projects with all dependencies!
- Support for forks, offline/ZIP-use
- Central repository like Maven Central
- Company-internal repositories required

Questions

- Use the PINF (package interface) object type?
  – But what about system-local naming?
- Versioning or no versioning?
- Integrated in abapGit or independent project?
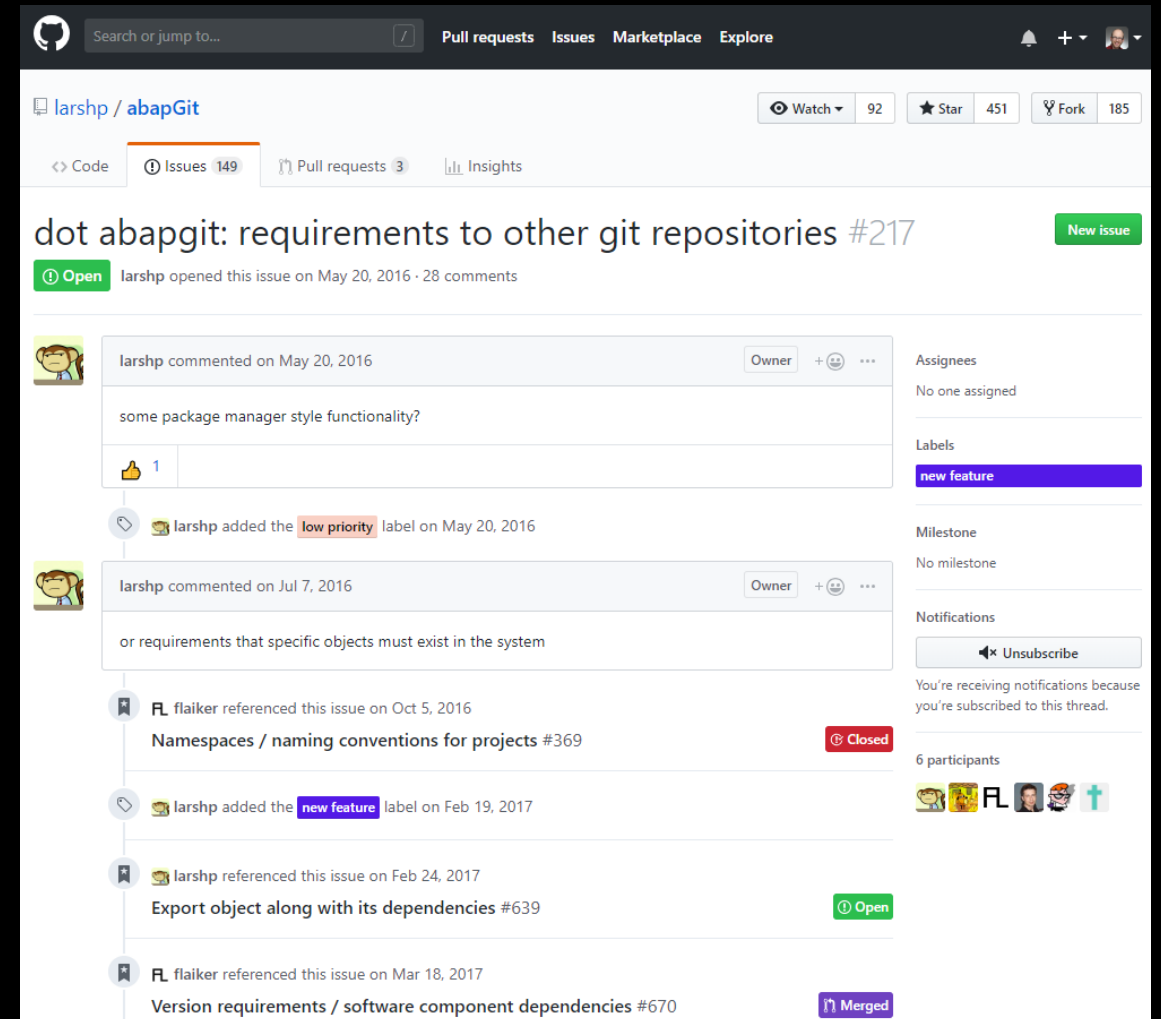
abapGit and Requirements – State-of-the-Art

- Requirements can be included optionally
- Coarse-grained requirement specification
  - Software Component
  - Minimum Release
  - Minimum Patchlevel/Service Pack
- Tight integration in abapGit

Let's welcome APACK!

- Independent of ABAP package concept (no PINF, DEVC…)
  – No issues with system-local names/missing global naming conventions
- Package definition for re-use (group ID, artifact ID, version)
- Explicit dependencies between re-use packages
- Fully compatible with abapGit, but designed independently

Rationale

- Steampunk whitelist might not grow fast enough
- Enable Steampunk users to develop and re-use „libraries"
- Nurture ABAP open-source community (dotabap.org)

Implementation – Status Quo

- Import Steampunk-only, integration in abapGit plugin for ADT
- Export On-Premise-only, integration in standard abapGit
- No versioning
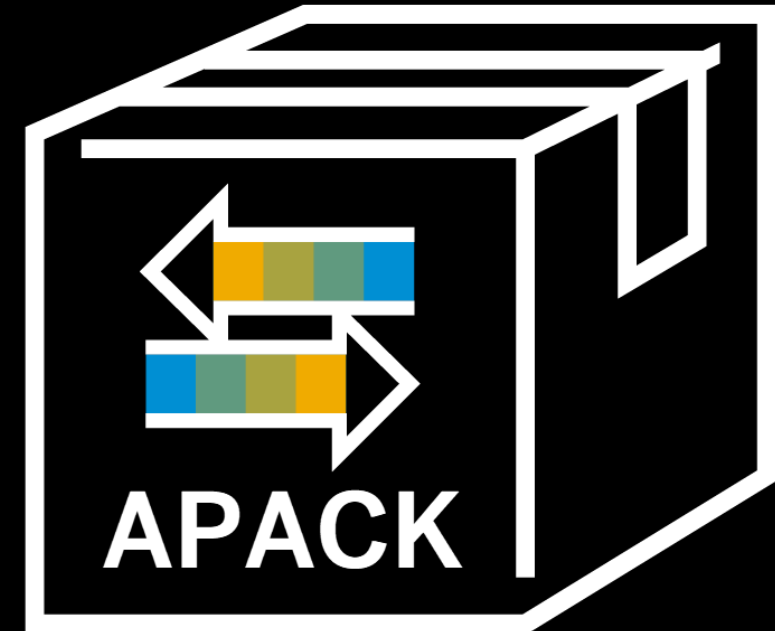
The APACK Manifest – ABAP implementation

- Marker interface IF_APACK_MANIFEST
- One class in a re-use library package must implement it
- Contains single descriptor field
  - Group ID
  - Artifact ID
  - Version
  - Repository Type (currently only ‚abapGit‘ and added automatically)
  - Git URL
  - Dependencies (Group ID, Artifact ID, Git URL)

```
  ZCL_JAK_APACK_MANIFEST ▶
 1⊖ CLASS zcl_jak_apack_manifest DEFINITION
 2    PUBLIC
 3    FINAL
 4    CREATE PUBLIC .
 5
 6    PUBLIC SECTION.
 7      INTERFACES: if_apack_manifest.
 8      METHODS: constructor.
 9    PROTECTED SECTION.
10    PRIVATE SECTION.
11  ENDCLASS.
12
13
14⊖ CLASS zcl_jak_apack_manifest IMPLEMENTATION.
15⊖   METHOD constructor.
16      if_apack_manifest~descriptor-group_id = 'sap.com'.
17      if_apack_manifest~descriptor-artifact_id = 'abap-platform-jak'.
18      if_apack_manifest~descriptor-version = '0.1'.
19      if_apack_manifest~descriptor-git_url = 'https://github.com/SAP/abap-platform-jak.git'.
20      if_apack_manifest~descriptor-dependencies = VALUE #( ( group_id    = 'sap.com'
21                                                              artifact_id = 'abap-platform-yy'
22                                                              git_url     = 'https://github.com/SAP/abap-platform-yy.git' ) ).
23    ENDMETHOD.
24
25  ENDCLASS.
```

The APACK Manifest – abapGit Serialization

- New descriptor file `.apack-manifest.xml`
- Located on top level of abapGit project
- Automated serialization via abapGit extension

SebastianWolf-SAP Add repository type to APACK manifest

📁 src	Add repository type to APACK manifest
📄 .abapgit.xml	Initialize JAK Repository
📄 .apack-manifest.xml	Add repository type to APACK manifest

SebastianWolf-SAP Add repository type to APACK manifest

1 contributor

20 lines (19 sloc) | 568 Bytes

```xml
1   <?xml version="1.0" encoding="utf-8"?>
2   <asx:abap xmlns:asx="http://www.sap.com/abapxml" version="1.0">
3    <asx:values>
4     <DATA>
5      <GROUP_ID>sap.com</GROUP_ID>
6      <ARTIFACT_ID>jak</ARTIFACT_ID>
7      <VERSION>0.1</VERSION>
8      <REPOSITORY_TYPE>abapGit</REPOSITORY_TYPE>
9      <GIT_URL>https://github.com/SebastianWolf-SAP/jak.git</GIT_URL>
10     <DEPENDENCIES>
11      <item>
12       <GROUP_ID>sap.com</GROUP_ID>
13       <ARTIFACT_ID>yy</ARTIFACT_ID>
14       <GIT_URL>https://github.com/SebastianWolf-SAP/yy.git</GIT_URL>
15      </item>
16     </DEPENDENCIES>
17    </DATA>
18   </asx:values>
19  </asx:abap>
```

Package Dependency Resolution during abapGit Import

- If APACK manifest is detected, all dependencies are automatically resolved
- Dependency resolution is done transitively, also dependencies of dependencies are taken in
- Local packages for all dependencies can be selected on a single wizard page
- Dependencies are all linked at once at wizard completion

APACK: ABAP Dependency Management made easy and convenient

- ABAP gets an automated package and dependency management for re-use libraries and applications
- Easy-to-use and almost no overhead
- Direct integration with abapGit
- Open Source as abapGit
- Independent project which can be easily extended, e.g.:
  - Central repository for sources/packages
  - Company-specific repositories
  - Versioned dependencies
  - Ports to on-premise ABAP releases
  - …
- Feedback and improvements welcome!
- HAVE FUN!
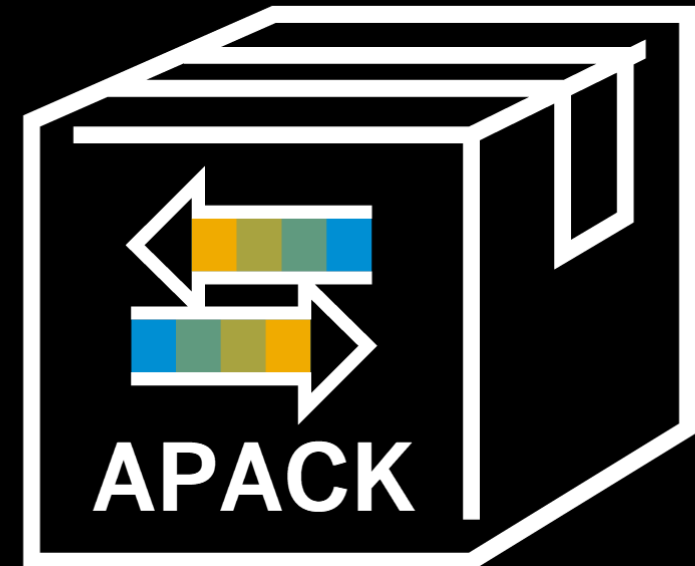
Thank you.

Contact information:

Sebastian Wolf
Development Architect – T&I Central Engineering
sebastian.wolf@sap.com
Twitter: @Ygriega, https://twitter.com/ygriega
GitHub: https://github.com/SebastianWolf-SAP
Blog Post: https://blogs.sap.com/2019/05/06/introducing-apack-a-package-and-dependency-manager-for-abap/