



## Development & Delivery of Enterprise Cloud Apps using the **SAP Cloud SDK**

PUBLIC

Marco Dahms, SAP Innovation Center  
May, 2019



THE BEST RUN 

# SAP Labs Berlin

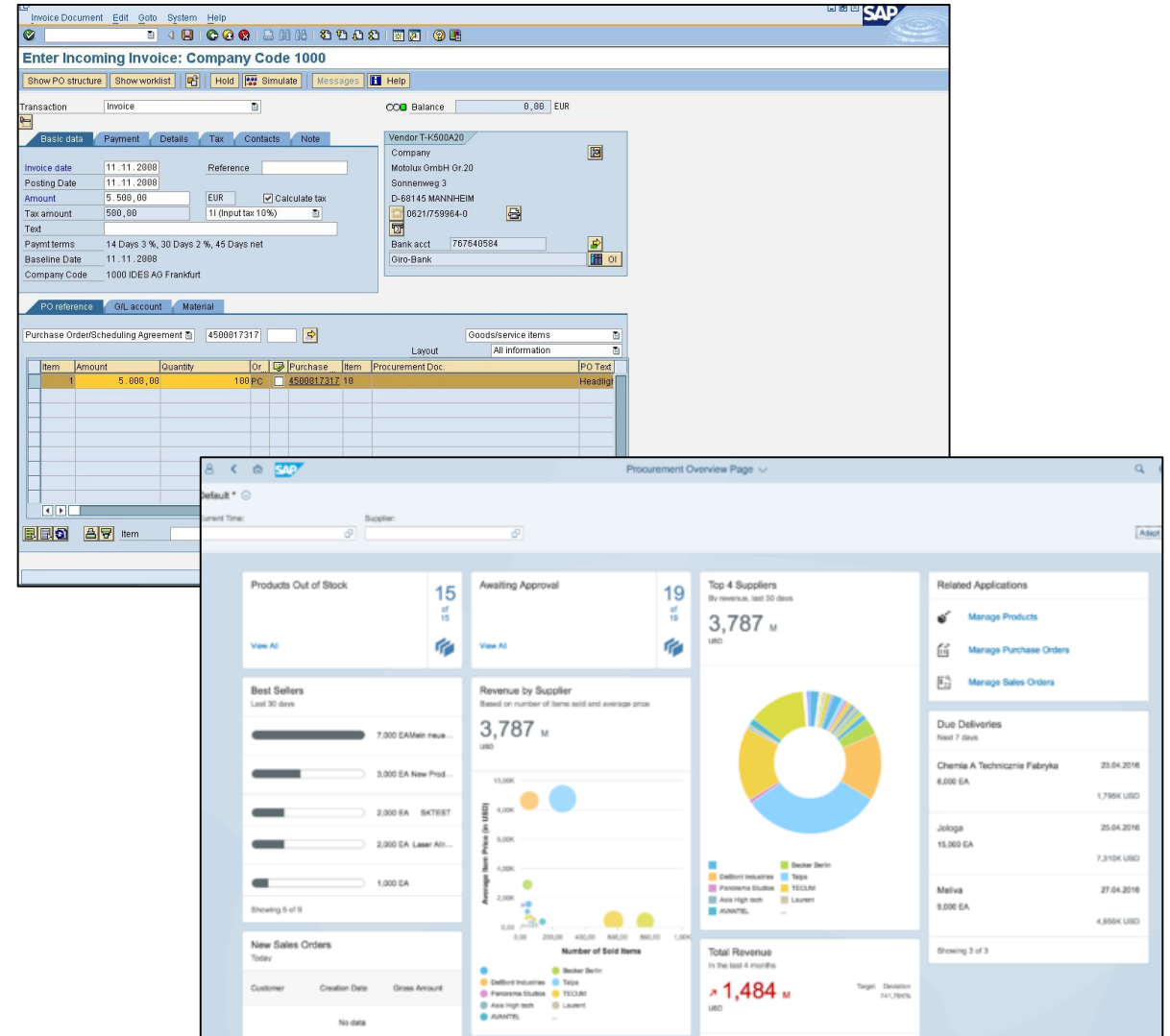
- Encompasses one lab in **Berlin** and SAP Innovation Center in **Potsdam**
- Established in 2013
- ~ **300** employees
- **We drive innovations at SAP**
  - Step-changing Cloud applications for businesses
  - Developer experience
  - Enablement of SAP's partners
  - Machine learning
  - Blockchain
  - ...





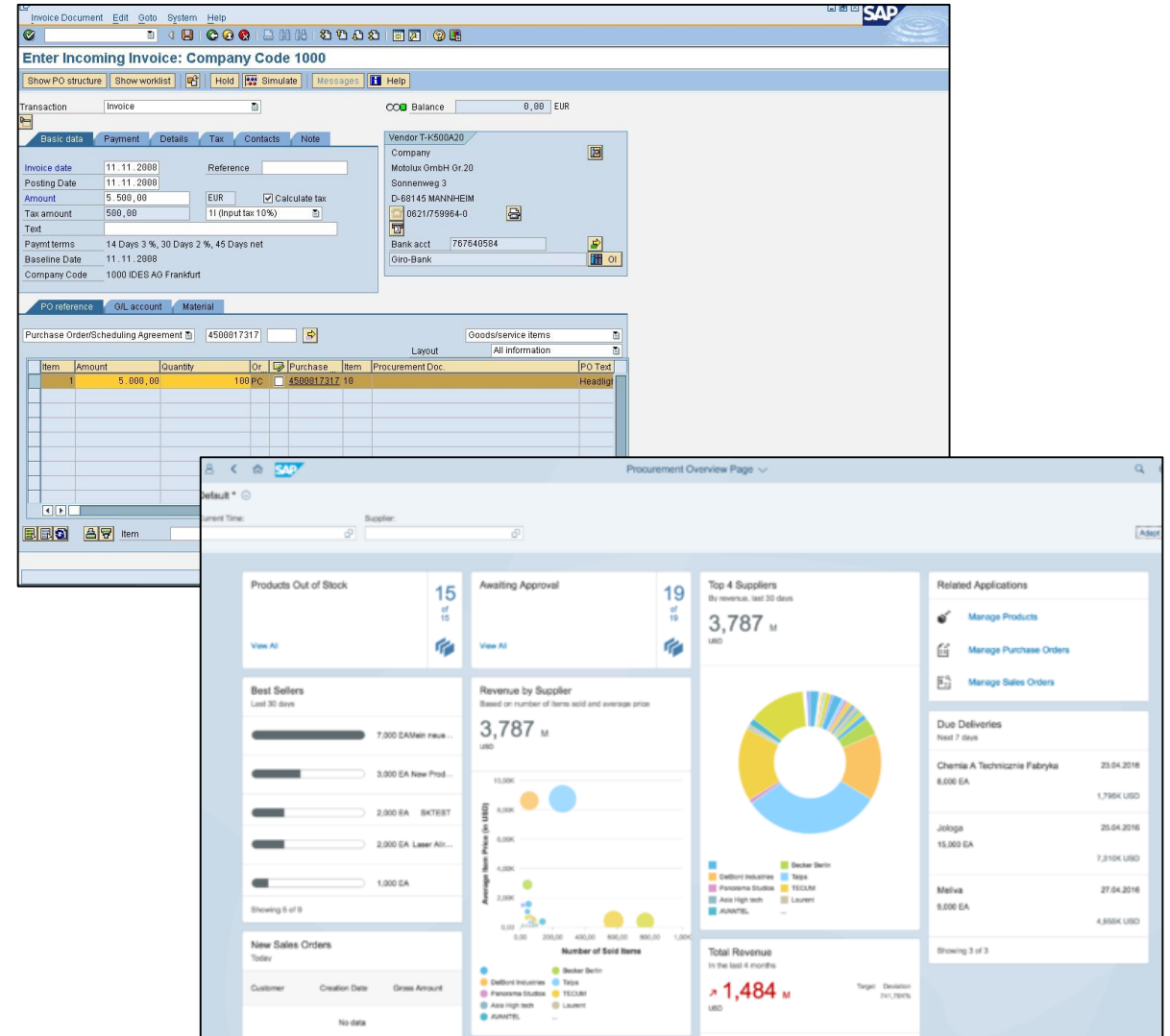
# “With what Software did SAP become successful?”

- Suite of highly-integrated **business apps**
  - Enterprise Resource Planning (ERP)
  - Financial Accounting
  - Warehouse Management
  - Sales & Distribution
  - Customer Relationship Management
  - **and many more**
- Product names you probably heard
  - SAP R/3
  - SAP ERP Business Suite
  - **SAP S/4HANA** (Cloud)
- **Majority** of our customers runs their business apps themselves **on-premise**

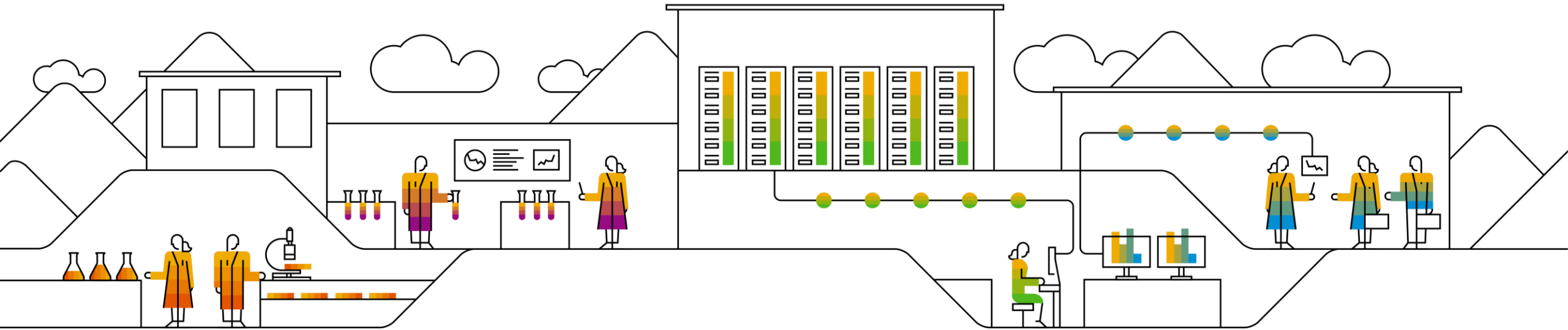


# Long-during innovation cycles in business software ecosystem

- Thinking about **daily delivery**...
- Customers quite **reluctant** to adopt new releases
  - 2 to 3 years** on average until software reaches end-user for **SAP ERP Business Suite**
- Reasons:
  - Risk of **many changes at once**
  - Testing across multiple landscapes
  - Very **conservative** rollout of changes



# How we deliver innovations more frequently



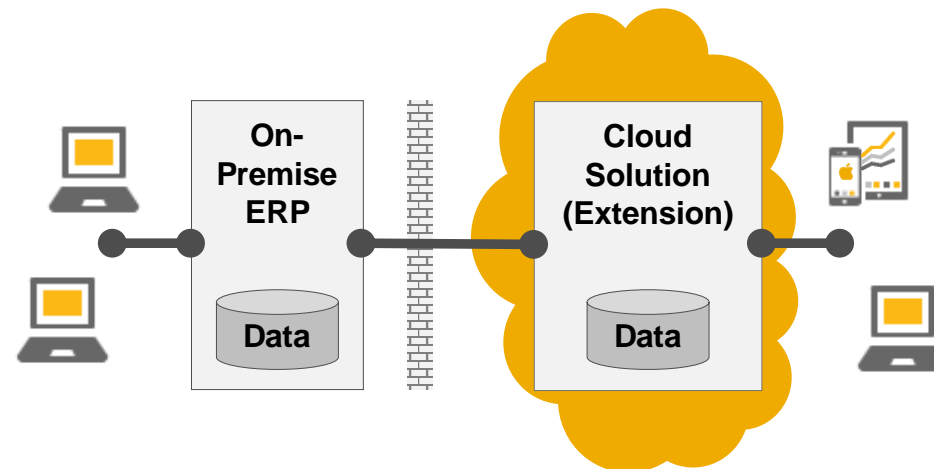
# Consume your business applications from SAP Cloud Platform

## New Solutions



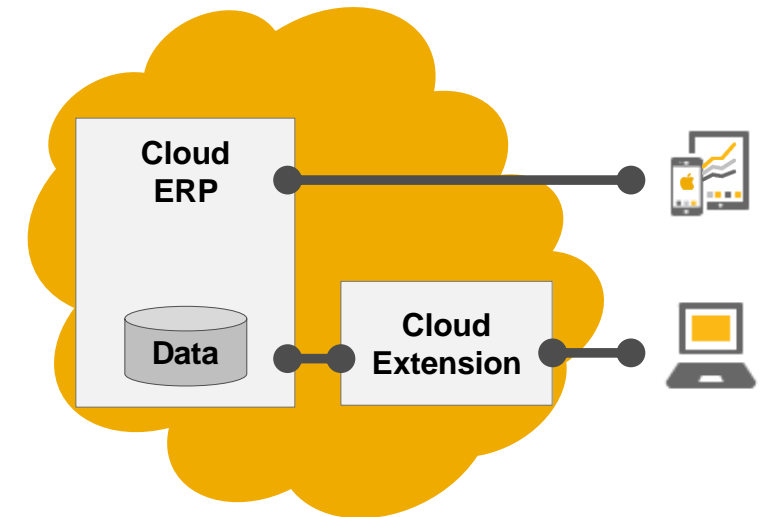
Build new stand-alone solutions running in the SAP Cloud Platform

## Hybrid Solutions



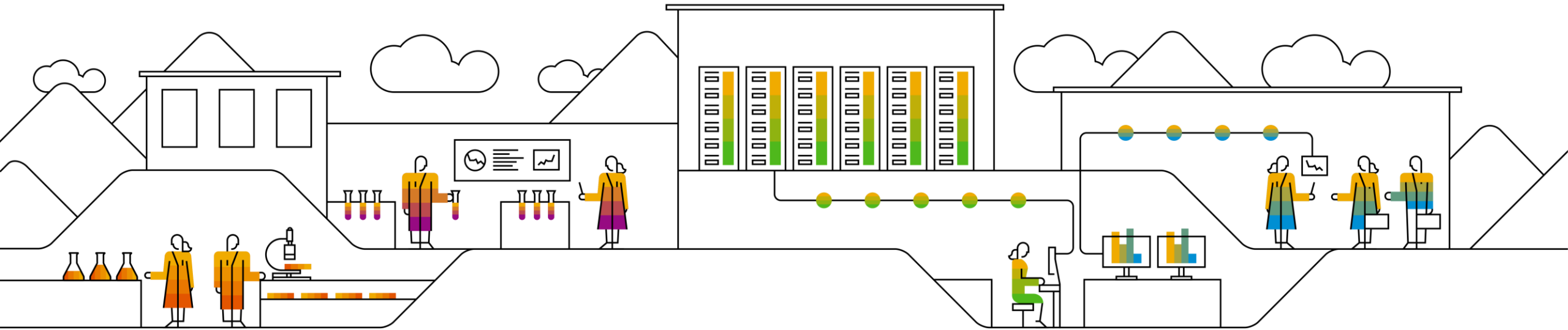
Extend existing on-premise solutions operated by the customer with new solutions running in the SAP Cloud Platform

## Cloud Extensions



Extend existing Cloud solutions with new solutions running also in the SAP Cloud Platform

# Show me that in action, please



# SAP RealSpend – A hybrid solution running in SAP Cloud Platform



App lifecycle independent from core ERP system



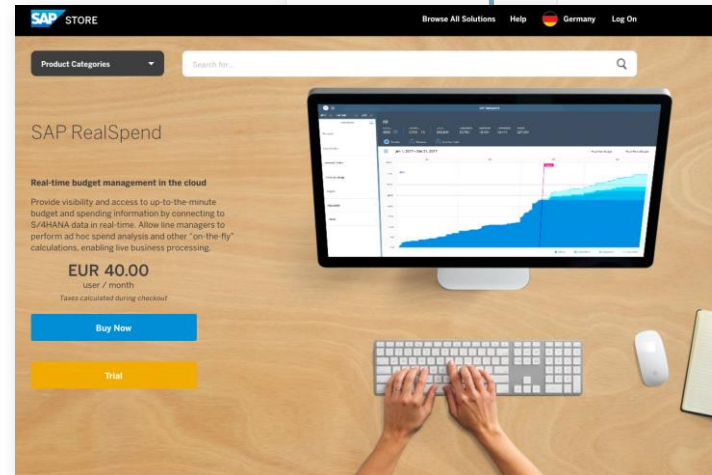
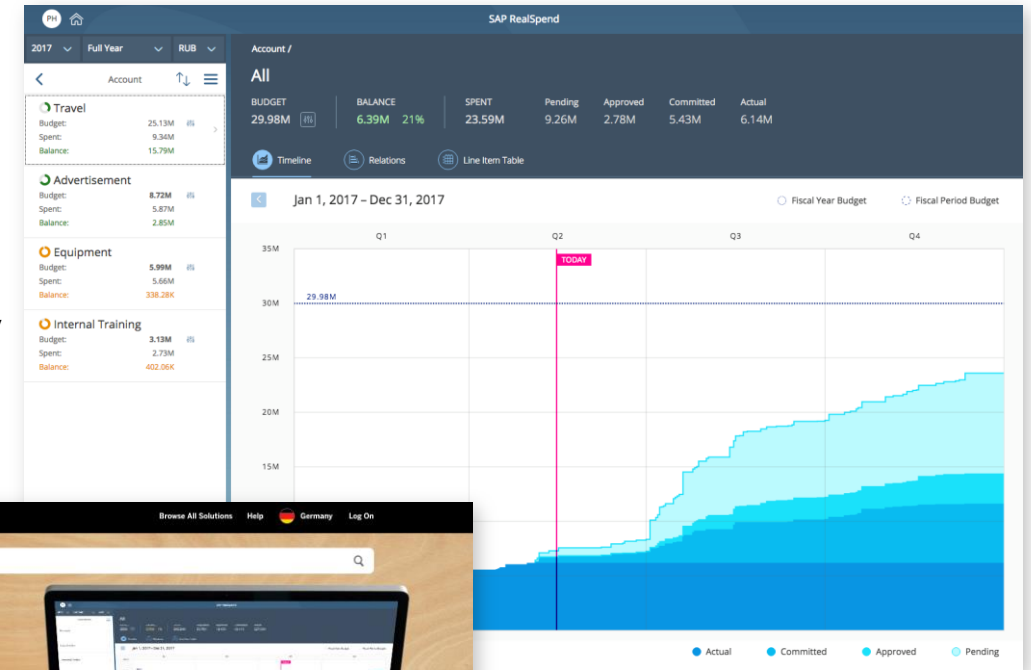
**Continuous Delivery:** Deployed multiple times per day



**No disruption** of core ERP system



Built on the shoulders of **open source**





## Our first build pipeline...



... we fixed it ...

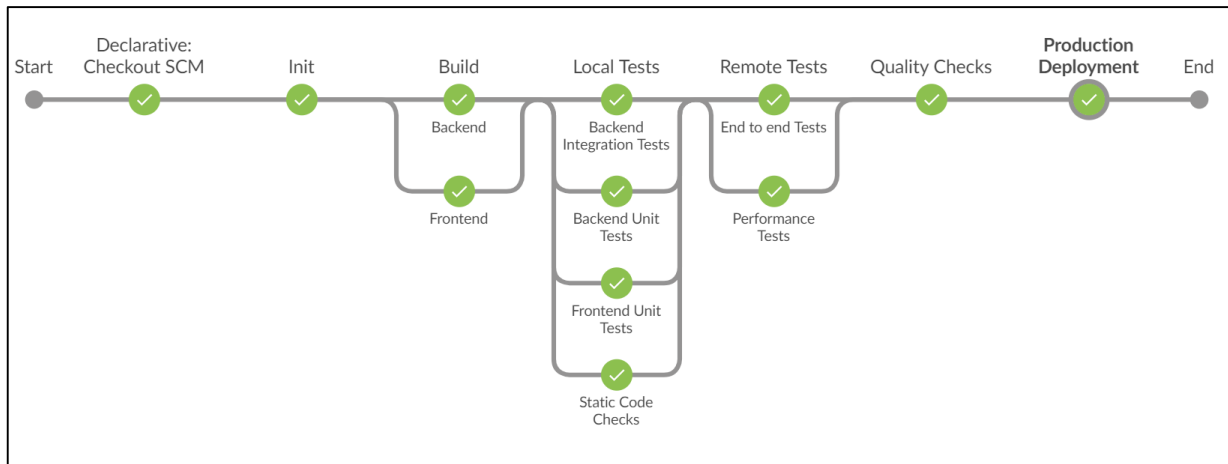


**... until we got it right!**



# Generalizing our learnings in a new development and continuous delivery toolkit

- **SAP Cloud SDK**
  - enables SAP developers, customers, and partners to develop applications on **SAP Cloud Platform**
- **Continuous Delivery Toolkit**
  - enables DevOps teams to ensure **high quality** with regards to performance, resource consumption, and operations



# SAP Cloud SDK for Java

The SAP Cloud SDK provides a very developer-friendly, **Java-based programming interface** with built-in cloud qualities to extend SAP S/4HANA on SAP Cloud Platform. Project templates, an out-of-the-box continuous delivery server, extensive documentation and a lively community create an attractive developer experience for partners and customers.



- Transparent integration with SAP S/4HANA Cloud and On-Premise
- Virtual Data Model for type-safe and fluent access to SAP S/4HANA in Java



- Eased platform transition (Neo > Cloud Foundry) due to abstractions for platform features
- Project templates for different execution environments



- Improved application performance by efficient caching management
- User and tenant specific caches



- Integration with resilience and fault tolerance framework
- Quality checks and test support to measure and ensure quality standards



- Simple integration of various extension frameworks for application development, such as toggling, persistence, etc.



- Out-of-the-box continuous delivery server
- Projects come with preconfigured continuous delivery pipeline including testing and checks

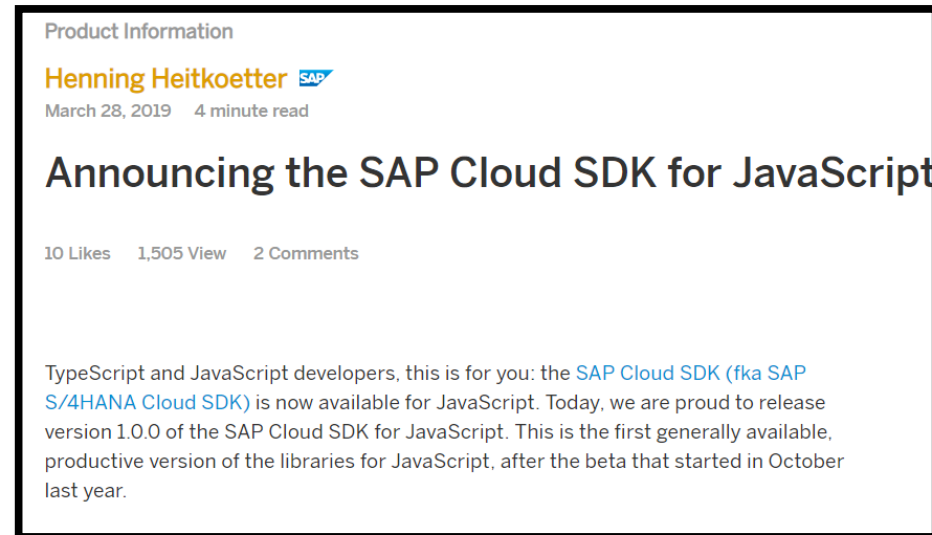
## Technology

- Java libraries for building of side-by-side extensions for SAP S/4HANA (on premise and cloud) **on SAP Cloud Platform**
- **Open Source** templates, scripts, and docker images for setting up **continuous integration pipeline and infrastructure in minutes**
- More than 30 tutorials, deep dives, and open source project examples demonstrating usage of the SDK. A **book will be released for Sapphire**



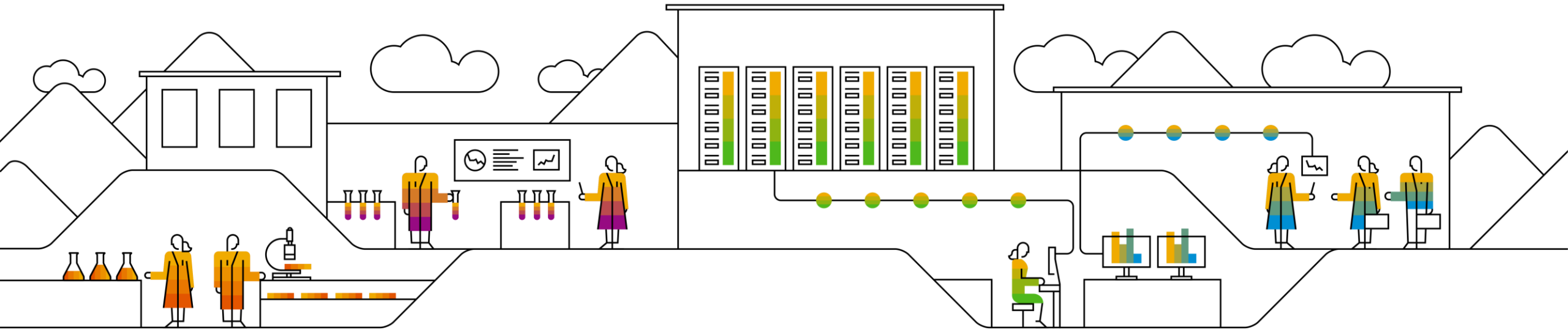
# SAP Cloud SDK for JavaScript

- SAP Cloud SDK for JavaScript
  - JavaScript Libraries (written in **TypeScript!**)
  - Continuous Delivery Toolkit (in progress)
- [Tutorials](#)
- [Documentation](#)

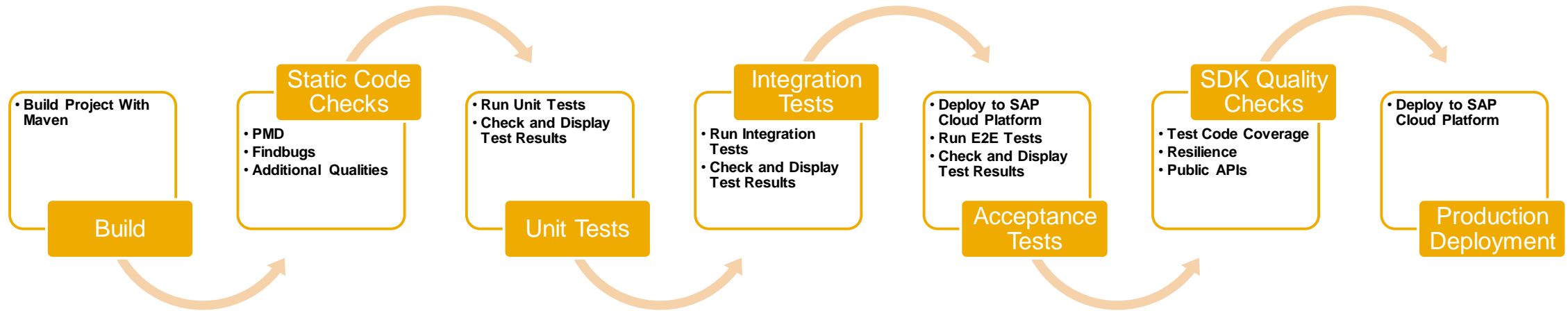


<https://blogs.sap.com/2019/03/28/announcing-the-sap-s4hana-cloud-sdk-for-javascript/>

# Show me that in action, please



# Ensuring High Quality with the Continuous Delivery Toolkit



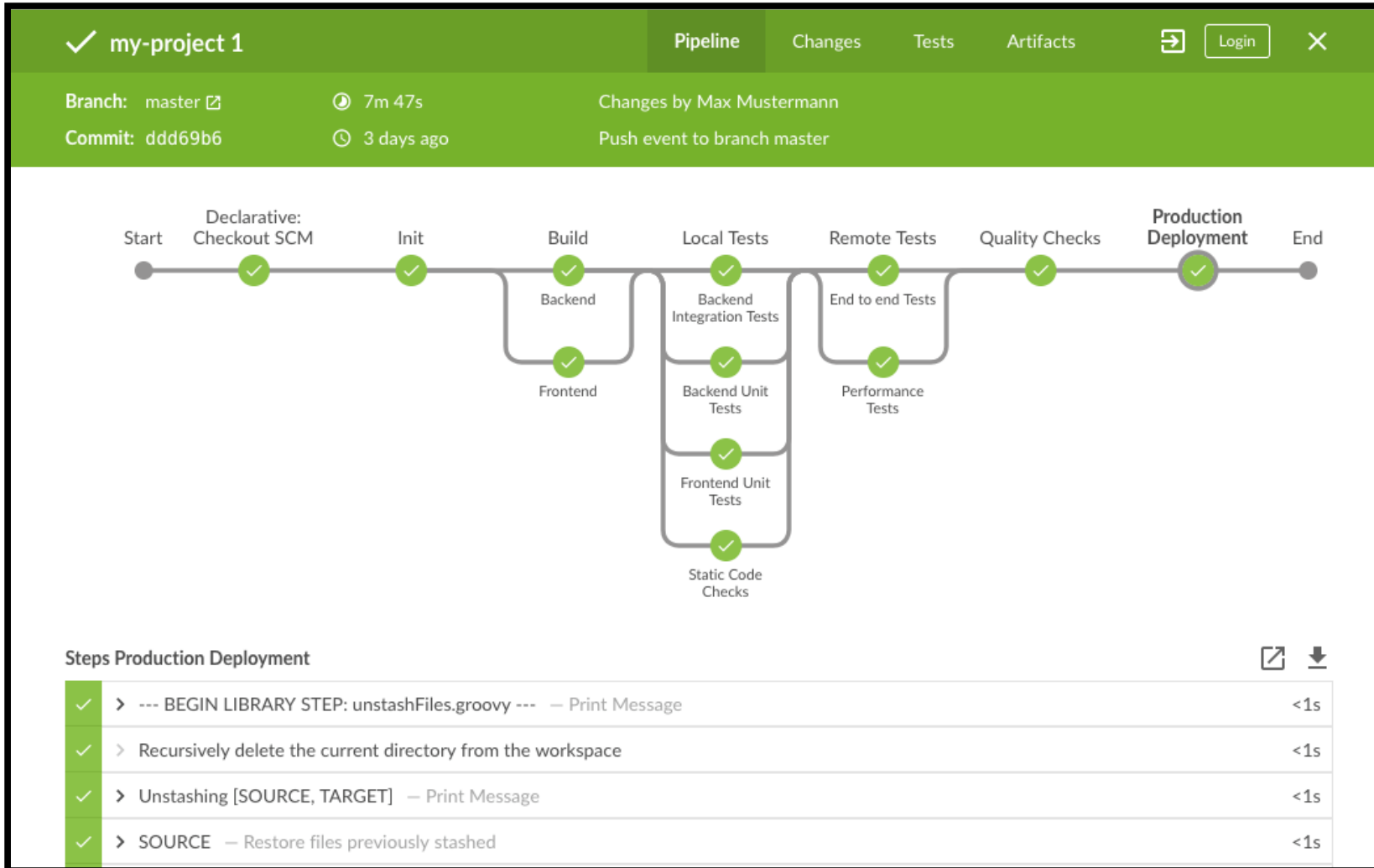
## Possible additional checks:

- Connect to Security Scanners
- Performance Tests

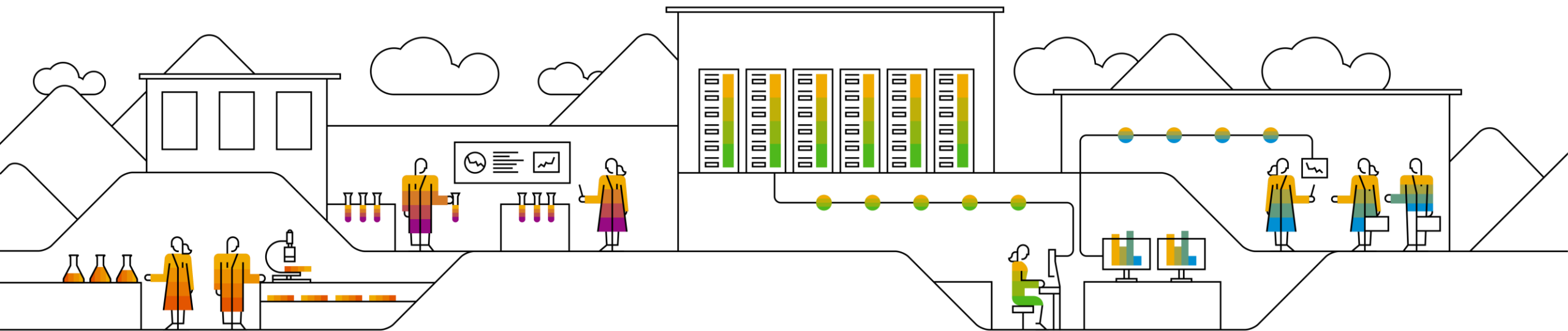
## Currently checked software qualities:

- Resilience: We check that every external call is wrapped into Hystrix
- API usage: We check that only whitelisted SAP APIs are used
- Post Mortem Analyzability: We check that exceptions are logged and the logger provided by the SDK is used

# Ensuring High Quality with the Continuous Delivery Toolkit



# How can **we** try that out?

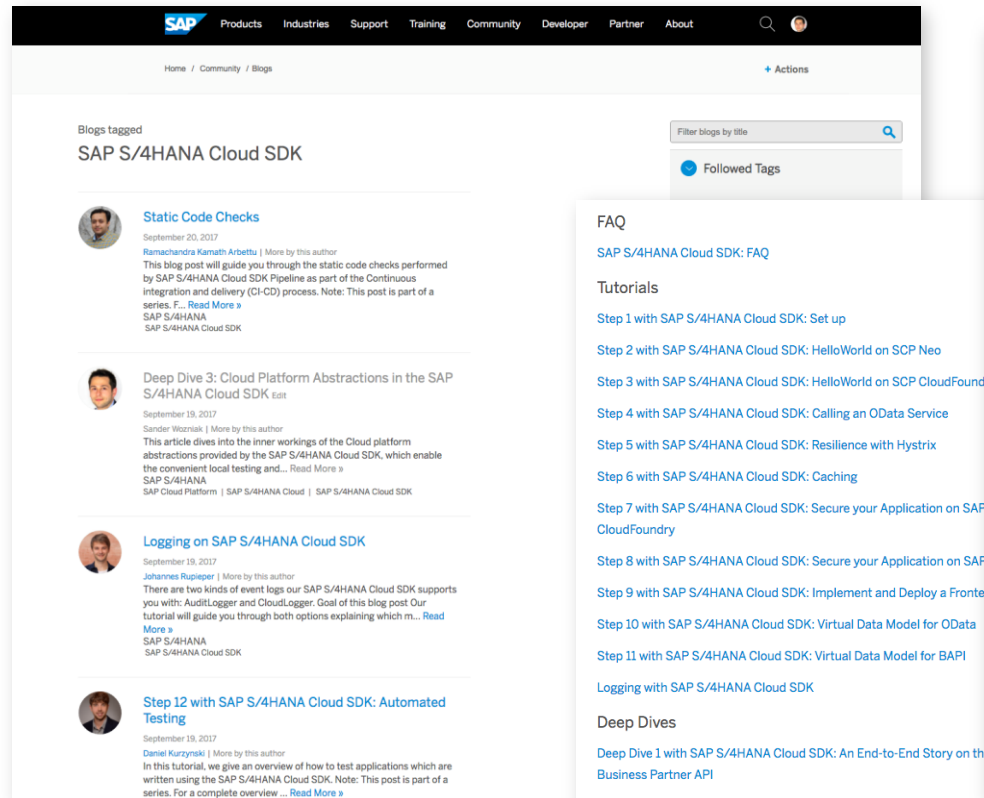




# Regular Blogs with Tutorials, Deep Dives, and FAQs

<https://blogs.sap.com/tag/sap-s4hana-cloud-sdk/>

<https://blogs.sap.com/2017/05/10/first-steps-with-sap-s4hana-cloud-sdk/>



## Concepts

Before we dive deeper into the real setup of the architecture, let's quickly review the architecture we intend to go for.

Figure 1 presents the final setup. First, we still have our existing "Hello World" or "Cost Center" Java-based microservice that we have created in the previous tutorials. However, instead of letting the customer access this application directly, we will use the so-called Application Router (App Router) that serves two purposes.

On the one hand, the App Router is a general entry point into the world of microservices. The main idea is that you can split an application into multiple microservices with independent deployability, polyglot runtimes & persistence and independent teams. Therefore, a central entry component is required that hides the complexity of the microservice landscape from the end customer.

On the other hand, the App Router is mainly responsible for managing authentication flows. The App Router takes incoming, unauthenticated requests from users and initiates an OAuth2 flow with the XSUAA service. The XSUAA service is an SAP-specific fork of CloudFoundry's UAA service to deal with authentication and authorization (it may again delegate this aspect to other providers such as external Identity Providers, see later in this tutorial). If the user authenticates at the XSUAA, it will respond with a JSON Web Token (JWT) containing the authenticated users as well as all scopes that he or she has been granted.

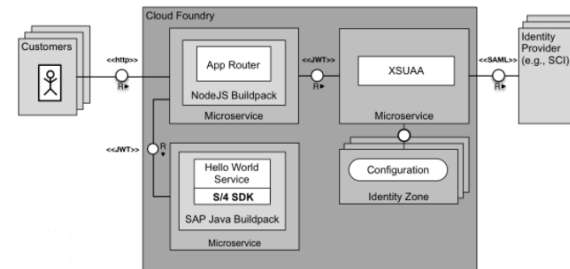



Figure 1: Authentication Flow during Runtime






## SAP S/4HANA Cloud SDK Overview

May 10, 2017 | 3,212 Views |

 Ekaterina Gavrilova  
more by this author

SAP S/4HANA

SAP S/4HANA Cloud SDK

 share 0  share 0  tweet  share 58  like 9

[Follow](#)

SAP S/4HANA has become de-facto ERP standard with now more than 5,800 customers worldwide. It accompanies organizations across the world in their digital transformation journey that requires them to adopt best-in-class software with high agility. This is further extended by SAP Cloud Platform that gives developers the power to build and run high-quality applications that conform to the highest order of performance, security and reliability tests.

This is where the SAP S/4HANA Cloud SDK comes in. Making the application development experience delightful, the SDK provides you out-of-the-box capabilities, such as an abstraction of the underlying cloud platform implementation (SCP Neo, Cloud Foundry), fault-tolerance, cache management, and tutorials and project templates.

You can access the SDK via the following resources:

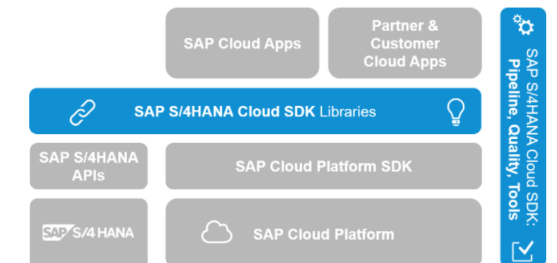
[SDK on Maven Central](#)

[Build pipeline](#)

[Build pipeline libraries](#)

[Dockerfiles for the SAP S/4HANA Cloud SDK pipeline](#)

[SDK Javadoc](#)



If you are interested to learn more, stay tuned for the upcoming development topics that we will

# Example Applications as Open Source on GitHub

<https://github.com/SAP/cloud-s4-sdk-examples>

The screenshot shows the GitHub repository page for `SAP / cloud-s4-sdk-examples`. The repository has 15 stars, 2 unstars, and 0 forks. It contains 18 commits, 2 branches, 0 releases, and 4 contributors. The repository is licensed under Apache-2.0. The main branch is `master`. The repository description states: "Runnable example applications that showcase the usage of the S/4HANA Cloud SDK." The repository contains the following files and folders: `Costcenter-Controller-CF` (updated 2 days ago), `.gitignore` (added 6 days ago), `LICENSE` (updated 6 days ago), `NOTICE` (created 6 days ago), and `README.md` (updated 2 days ago). The `README.md` file is displayed, showing the title "SAP S/4HANA Cloud SDK - Examples and Documentation" and the description "Runnable example applications that showcase the usage of the SAP S/4HANA Cloud SDK." The description also states: "The SAP S/4HANA Cloud SDK helps to develop SAP S/4HANA extension application on the SAP Cloud Platform. This repository contains example projects that demonstrate, how developers can use SAP S/4HANA Cloud SDK components to build extensions for SAP S/4HANA on SAP Cloud Platform. The following applications are included:"

- Cost Center Manager App
- Employee Browser (Integration with S/4HANA and SuccessFactors)

The screenshot shows the GitHub repository page for `SAP / cloud-s4-sdk-examples`, displaying the code for `CostCenterController.java`. The repository has 15 stars, 2 unstars, and 0 forks. It contains 18 commits, 2 branches, 0 releases, and 4 contributors. The repository is licensed under Apache-2.0. The main branch is `master`. The repository description states: "Runnable example applications that showcase the usage of the S/4HANA Cloud SDK." The repository contains the following files and folders: `Costcenter-Controller-CF` (updated 2 days ago), `.gitignore` (added 6 days ago), `LICENSE` (updated 6 days ago), `NOTICE` (created 6 days ago), and `README.md` (updated 2 days ago). The `CostCenterController.java` file is displayed, showing the following code:

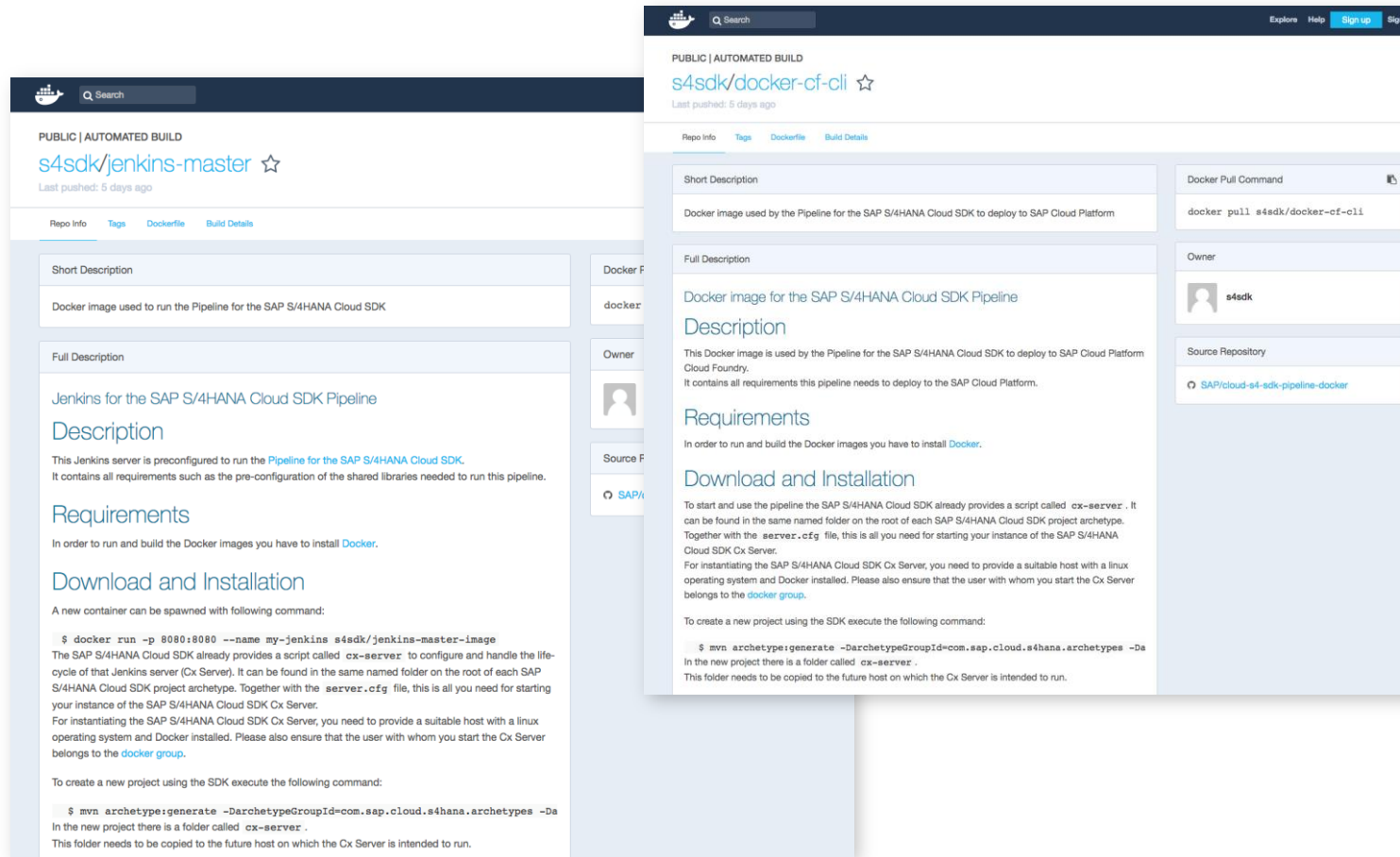
```
1 package com.sap.cloud.sdk.tutorial.controllers;
2
3 import org.slf4j.Logger;
4 import org.springframework.http.HttpStatus;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RequestBody;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.bind.annotation.RequestParam;
11 import org.springframework.web.bind.annotation.RestController;
12
13 import java.util.List;
14 import java.util.Locale;
15
16 import com.sap.cloud.sdk.cloudplatform.logging.CloudLoggerFactory;
17 import com.sap.cloud.sdk.s4hana.connectivity.ErpConfigContext;
18 import com.sap.cloud.sdk.s4hana.connectivity.ErpDestination;
19 import com.sap.cloud.sdk.s4hana.datamodel.bapi.structures.ReturnParameter;
20 import com.sap.cloud.sdk.s4hana.serialization.SapClient;
21 import com.sap.cloud.sdk.tutorial.command.CreateCostCenterCommand;
22 import com.sap.cloud.sdk.tutorial.command.GetCostCenterCommand;
23 import com.sap.cloud.sdk.tutorial.models.CostCenterDetails;
24
25 @RestController
26 public class CostCenterController
27 {
28     private static final Logger logger = CloudLoggerFactory.getLogger(CostCenterController.class);
29
30     private ErpConfigContext getErpConfigContext( final String sapClient ){
31         final ErpConfigContext config = new ErpConfigContext(
32             ErpDestination.getDefaultName(),
33             new SapClient(sapClient),
34             Locale.ENGLISH);
35         return config;
36     }
37 }
```

<https://search.maven.org/#search%7Cga%7C1%7Ccom.sap.cloud.s4hana>

24

# SDK Jenkins Pipeline on Docker Hub

<https://hub.docker.com/u/s4sdk/>



# Pipeline Library and Code available as Open Source on GitHub

<https://github.com/SAP/cloud-s4-sdk-pipeline>

<https://github.com/SAP/cloud-s4-sdk-pipeline-lib>

This screenshot shows the GitHub repository page for `SAP / cloud-s4-sdk-pipeline`. The repository has 16 stars, 7 forks, and 0 issues. It contains 12 commits, 1 branch, and 0 releases. The repository is licensed under Apache-2.0. The commit history shows a series of updates to the pipeline, including updates to the LICENSE, NOTICE, README.md, build.gradle, and greclipse.properties files. The README.md file is displayed below the commit history, showing a diagram of the pipeline for the SAP S/4HANA Cloud SDK.

**Pipelines for the SAP S/4HANA Cloud SDK**

The diagram illustrates the pipeline flow for the SAP S/4HANA Cloud SDK. It starts with a 'Start' node, followed by 'Declarative: Checkout SCM', 'Init', 'Build', 'Local Tests', 'Remote Tests', 'Quality Checks', 'Production Deployment', and finally 'End'. The 'Build' step is further detailed with sub-steps: 'Backend', 'Frontend', 'Backend Integration Tests', 'Backend Unit Tests', 'Frontend Unit Tests', and 'Static Code Checks'. The 'Local Tests' step includes 'End to end Tests' and 'Performance Tests'.

This screenshot shows the GitHub repository page for `SAP / cloud-s4-sdk-pipeline-lib`. The repository has 16 stars, 6 forks, and 0 issues. It contains 12 commits, 1 branch, and 0 releases. The repository is licensed under Apache-2.0. The commit history shows a series of updates to the pipeline, including updates to the LICENSE, NOTICE, README.md, build.gradle, and greclipse.properties files. The README.md file is displayed below the commit history, showing a description of the pipeline library for the SAP S/4HANA Cloud SDK.

**Pipeline Library for the SAP S/4HANA Cloud SDK**

**Description**

This pipeline library is used by [Pipeline for the SAP S/4HANA Cloud SDK](#). It defines the common steps (functions) in a Jenkins pipeline to build, test and deploy applications.

**Requirements**

To use the pipeline library you must have a git project which uses a pipeline, such as the [Pipeline for the SAP S/4HANA Cloud SDK](#).

**Download and Installation**

To use the library in a pipeline you have to configure this library as [global shared library](#).

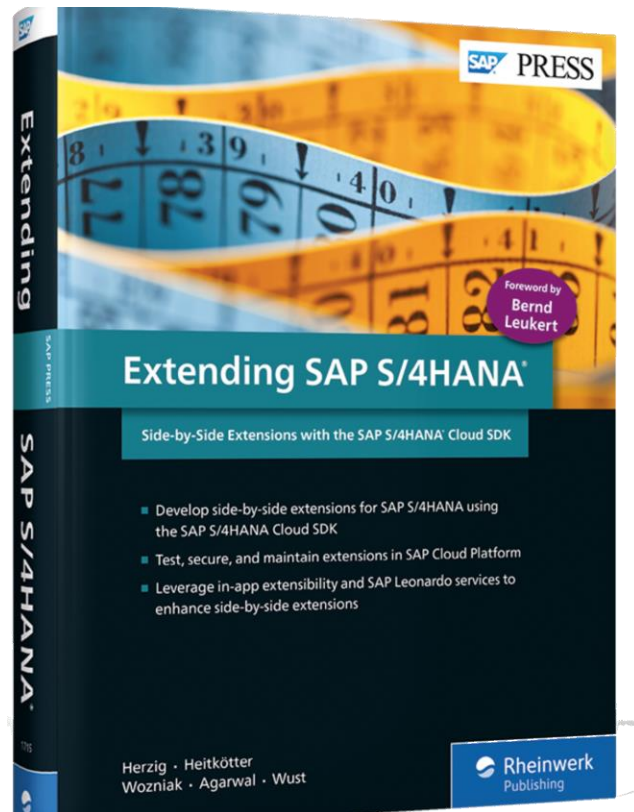
To avoid the manual steps described in that documentation you can use the SAP S/4HANA Cloud SDK Cx Server.



# More Information

## S/4HANA Extensibility

### Book



### Link Collection

[Extensibility Explorer](#)

[Custom Code Adaptations for S/4HANA](#)

[Custom Code Migration Worklist](#)

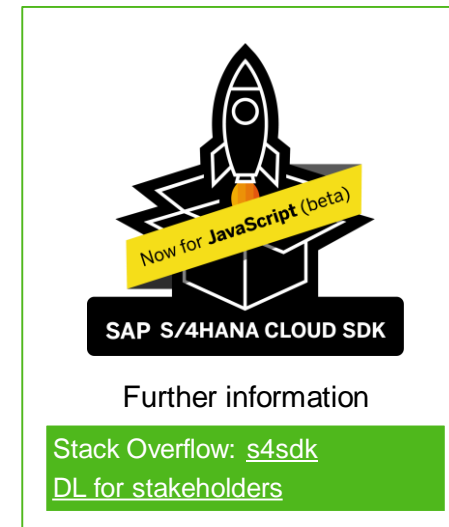
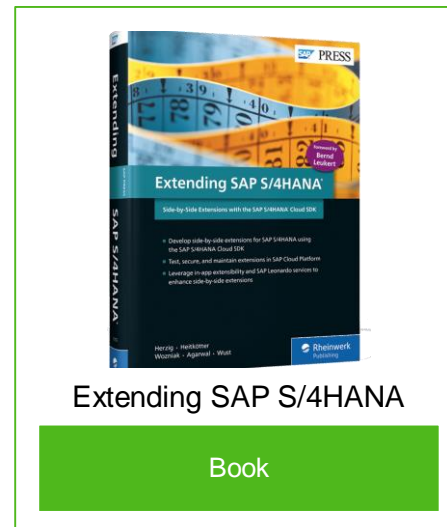
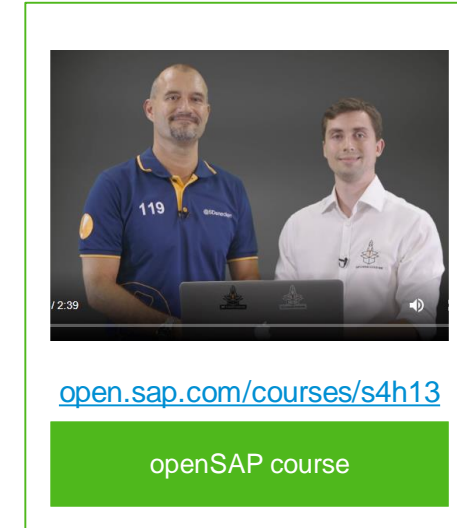
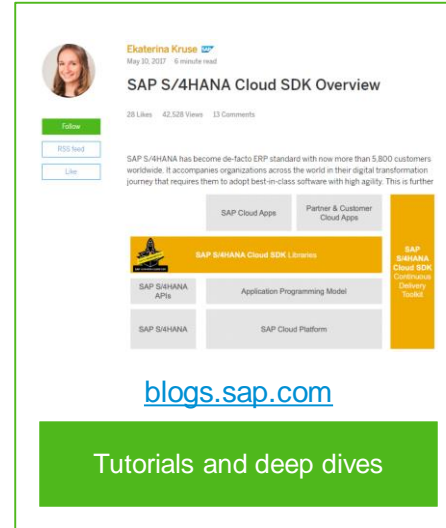
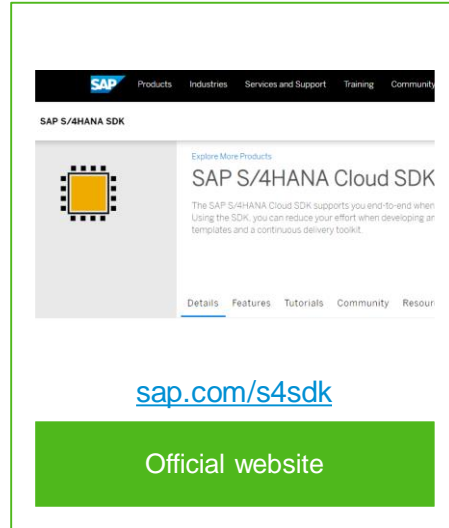
[SAP S/4HANA Cloud SDK](#)

[Side-by-side examples \(Open Source\)](#)

[Continuous Delivery Toolkit \(Open Source\)](#)

[More examples \(Open Source\)](#)

# Great documentation is one click away



# Thank you.

Contact information:

**Marco Dahms**

Senior Software Developer

SAP Labs Berlin, Germany

SAP Innovation Center Potsdam, Germany

[marco.dahms@sap.com](mailto:marco.dahms@sap.com)