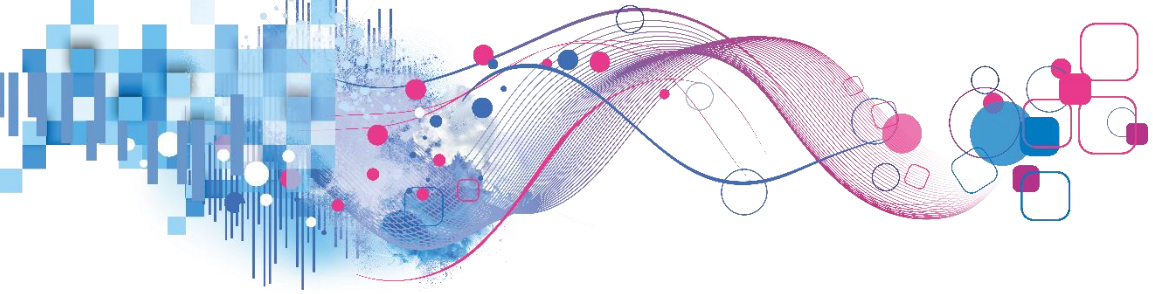
Pipeline as a Service

Project NIKE

Sprint 9 – Demo

30th December 2019

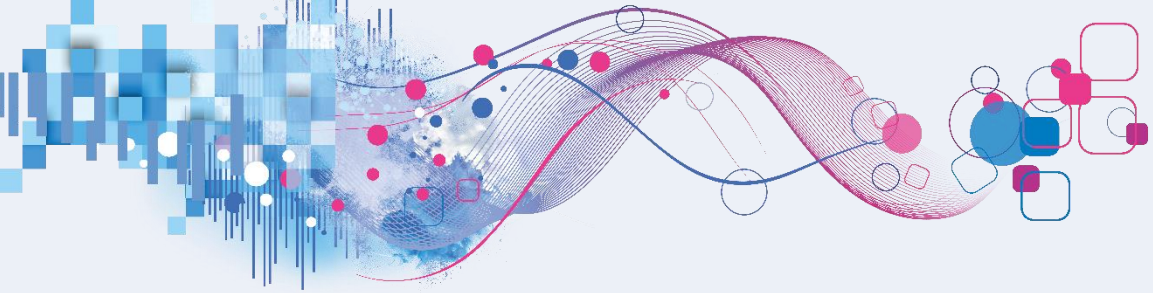
experian™



Agenda

- Sprint 9 updates
- Features completed and increments
- Demo
- Feedback

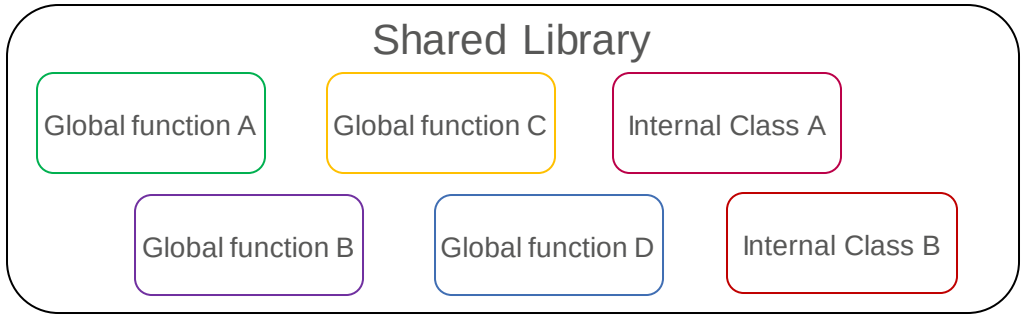
experian™

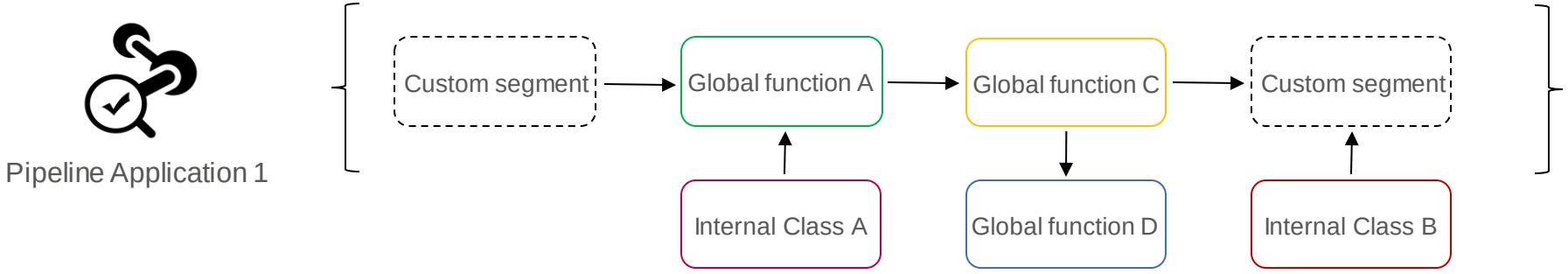


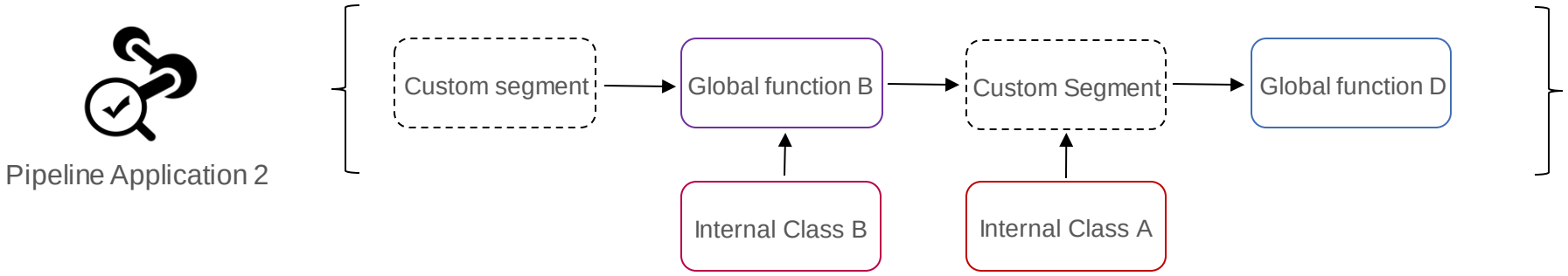
Features completed and increments

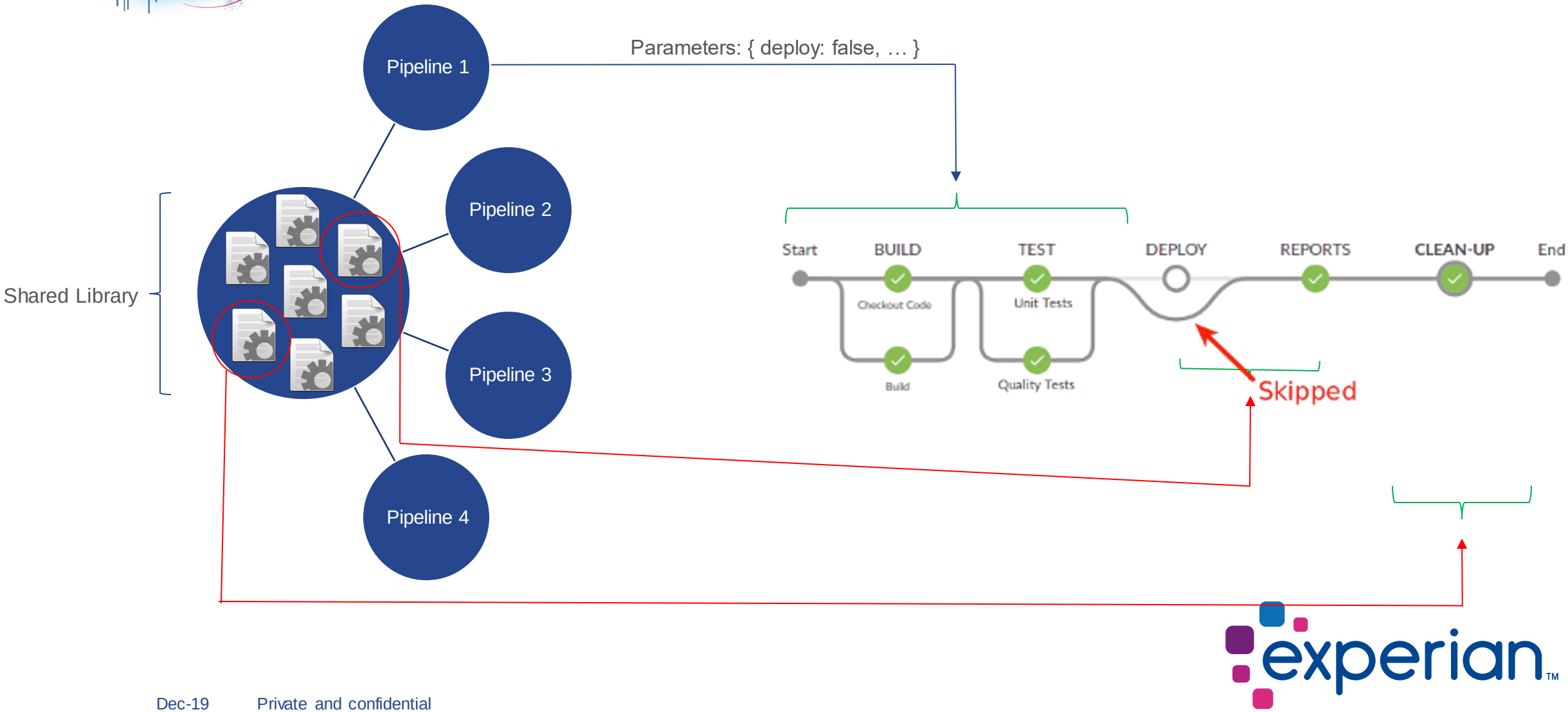
- Global DA GSG Pipeline discovery (Phase 2)
- Colombia CI/CD Pipeline discovery (Phase 1)
- UK PH "Regional Business Functions Solutions" Pipeline Discovery (Phase 1)
- Cucumber Support in the pipeline
- Jenkins Stage Upgrade

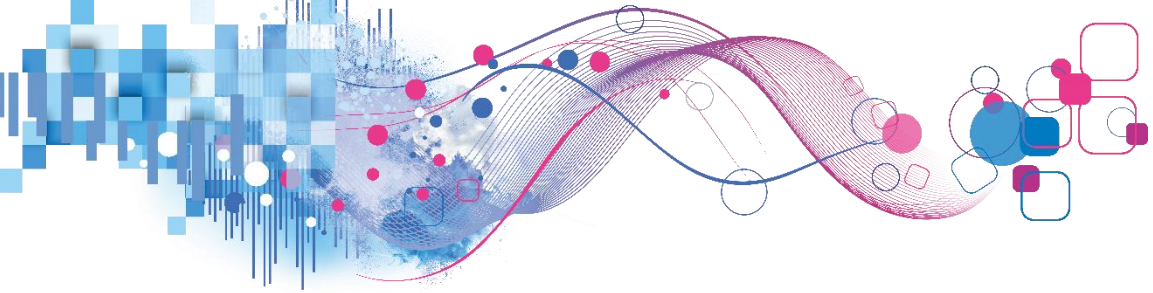
EDNK-259


Shared Library
Global function A
Global function C
Internal Class A
Global function B
Global function D
Internal Class B


Pipeline Application 1
Custom segment
Global function A
Global function C
Custom segment
Internal Class A
Global function D
Internal Class B


Pipeline Application 2
Custom segment
Global function B
Custom Segment
Global function D
Internal Class B
Internal Class A

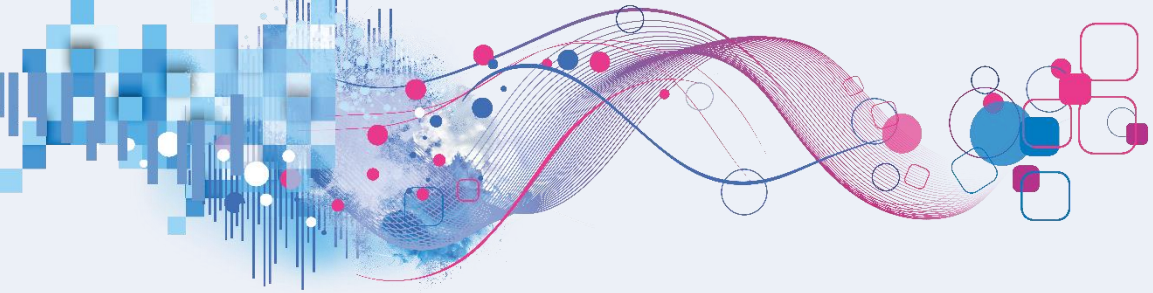

Pipeline 1
Parameters: { deploy: false, … }
Pipeline 2
Shared Library
Pipeline 3
Pipeline 4
Start
BUILD
TEST
DEPLOY
REPORTS
CLEAN-UP
End
Checkout Code
Build
Unit Tests
Quality Tests
Skipped



Identified global function approaches:

1. **Library**
   - Defined global variables with structure that provide specific functionality (E.g.: build, Unit testing, create releases, etc.)
   - Essentially, this global variables are just externalized scripts that can be imported into Jenkinsfiles to break down logic, providing common functionality across pipelines, following the DRY principles.
   - Usually those scripts contains a few internal methods and/or variables in order to provide specific required functionality
2. **Pipeline**
   - Defined global variables that extends stages and capabilities for pipelines
   - As with the library approach, this global variables are just externalized scripts but with the difference that they manage logic (sometimes complex logic) to provide additional stages to pipelines
   - The flow that will follow the script is controlled by arguments provided when performing the main call. This requires a fully understanding of the script for its reference to be successful

For both approaches, another global function can be call from a specific global function



EDNK-260 - CLONE - Understand and document infrastructure as code automation in UK DA GSG pipeline

1. All infrastructure are managed by **Packer** to create new images, **ansible** to provision configurations and **terraform** to deploy infrastructure

   Infrastructure traceable and versioned - Terraform modules are not re-used

2. Jenkins slaves are docker containers well defined as code.

   One slave for each technology (maven, packer, npm, terraform etc)

3. Lot of Ops tasks are defined on several repositories as bash scripts, powershell scripts, ansible playbooks, and some chef recipes

4. Infrastructure are using a few Jenkins shared libraries defined on platform-cicd-pipeline-library most commons: buildDockerImage, buildDockerImageWithOpenshift and buildDeployPipeline and

https://confluenceglobal.experian.local/confluence/display/EPS/Platform

EDNK-256 - Discovery - UK PH - "Regional Business Functions - Solutions" Pipeline (Phase 1)

Findings

1. Custom library functions from almost everything: from simple shell executions to complex OpenShift deployments.
2. **Tightly coupled** code.
   - Maintainability
   - Development
   - Testing
   - It is complex.
3. Very robust solution that meets their needs but could be hard to adopt for other projects.

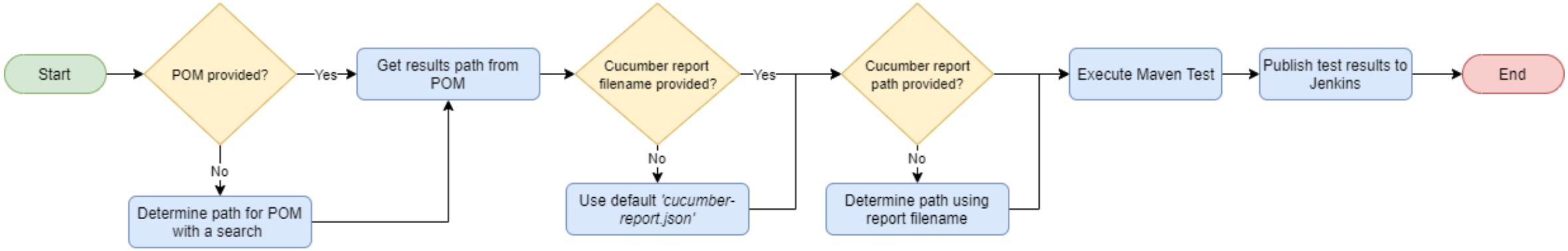
EDNK-239 Cucumber Support

- A Shared Library was developed

EITS Jenkins / devopspipeline-library

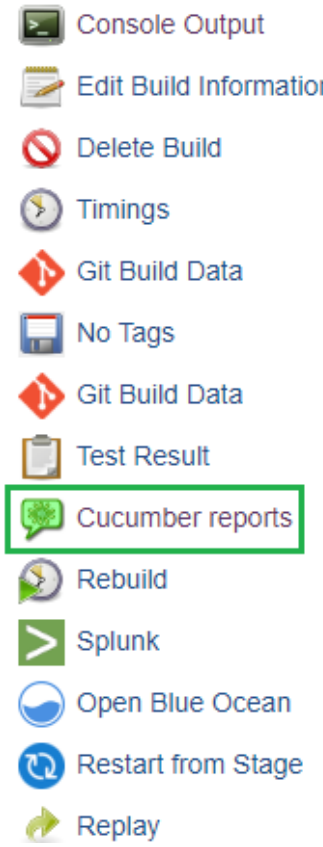
Source

develop ··· | devopspipeline-library / vars / **devopsBuildAndTestWithJUnitCucumber.groovy**



EDNK-239 Cucumber Support

- The pipeline can successfully run Cucumber tests.
- the build and deploy process will fail if the tests do not pass.
- A notification is sent upon failure of the tests
- Validate that the test coverage report is available after the tests have executed successfully.



Feature Report

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
Hello message	4	0	0	0	0	4	1	0	1	0.279	Passed

Feature Hello message

All users will receive a hello message upon running the Hello application

Background 0.006

Steps

Given a user, whoever he might be 0.006

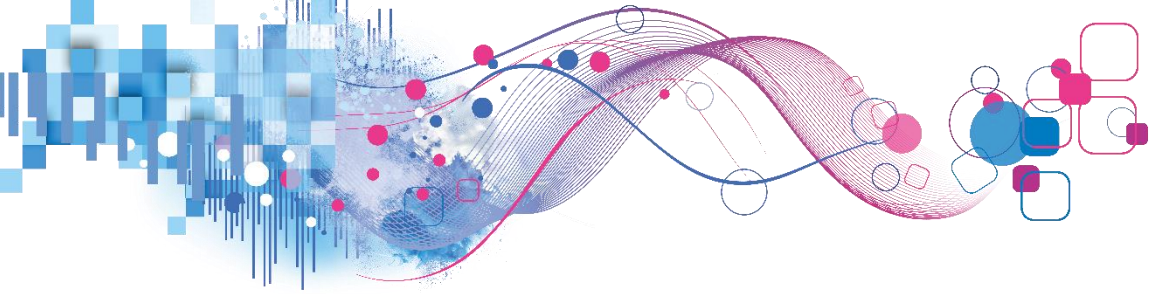
Scenario The user runs the application 0.273

Hooks

Before `CucumberSpringContextConfiguration.setUp()` 0.003

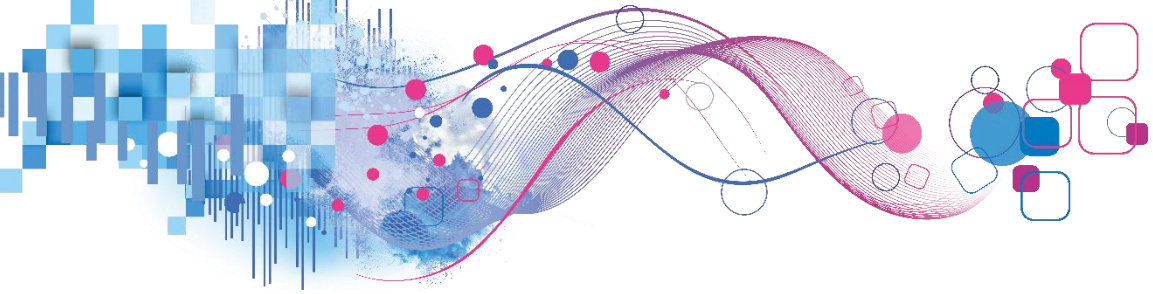
Steps

Given I enter the correct hostname for the application 0.001
When I wait for a response 0.000
Then I should see a "**Greetings from Spring Boot!**" message 0.272



EDNK-246 Jenkins Operation Center Temp Env Upgrade

- Cloned Current Jenkins Operation Center Env to Temp Env (except jobs)
- Resolved JOC parameters to new temp IP's
- Jenkins Operation Center has been upgraded to 2.190.3.2
- Client Masters DEV-MASTER-TEMP and ARM-MASTER-Temp Upgraded to 2.190.3.2
- Updated plugins to current JOC versions

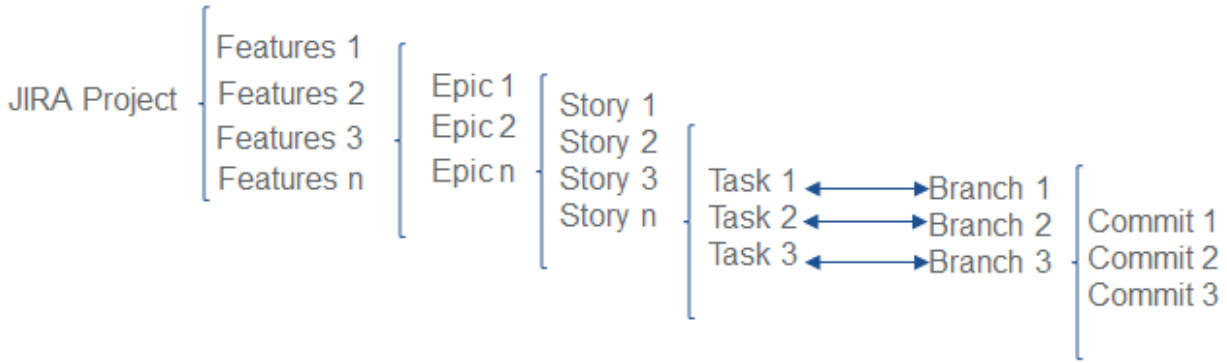


EDNK-255 Colombia CI/CD Pipeline discovery (Phase 1)

Jenkins
2.206 Cloudbees version

Colombia CI/CD Pipeline Implementation

SDLC / SAFE Implementation



Notes

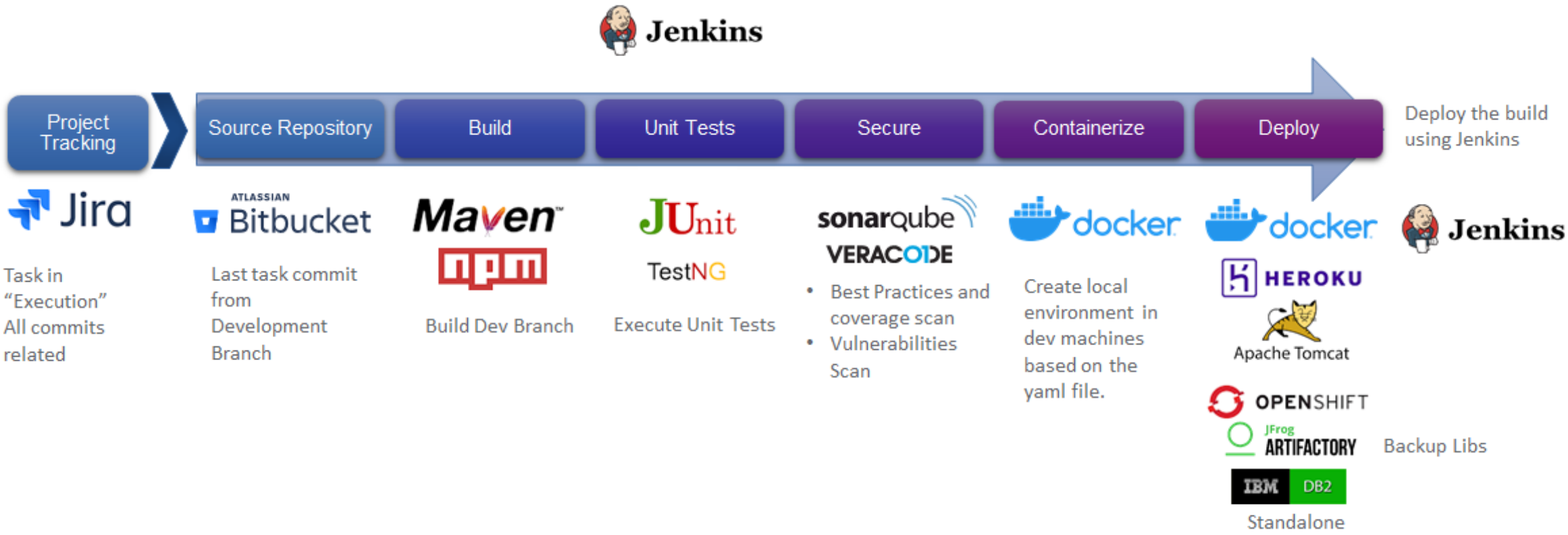
64 Projects:

- Until 3 software developments per project.
- 16 teams.

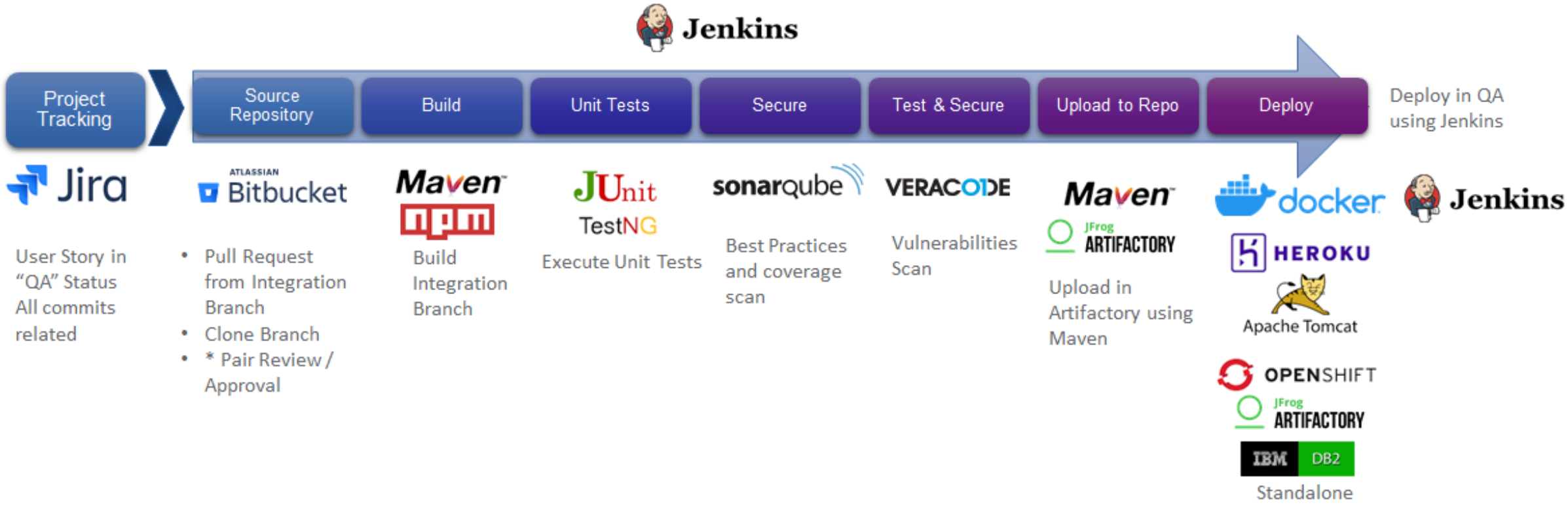
JIRA is integrated with Bitbucket.
Pipelines are triggered from JIRA (There is a button in the Epic)

•Each Software component is versioned in Bitbucket as an independent repository; a branch is needed per repository.
•A task may have related several commits (changes) as well, as several branches (repositories), as development requires.
•Commits applied to tasks will be deployed in Dev environment; In case requirement is to deploy to Tests then, execute a Pull Request to the integration branch and, then this commit must be related to a (SAFe) Story.
•After resolving the pull request, the dev branch will disappear for these branches are temporary.
•In case you need to modify temp libraries, you must first execute the pull request before running dev pipeline, so that these, will be automatically installed in Artifactory.

Dev Pipeline



QA (Integration Env) Pipeline



- User Story in "QA" Status
- All commits related

- Pull Request from Integration Branch
- Clone Branch
- * Pair Review / Approval

Build Integration Branch

Execute Unit Tests

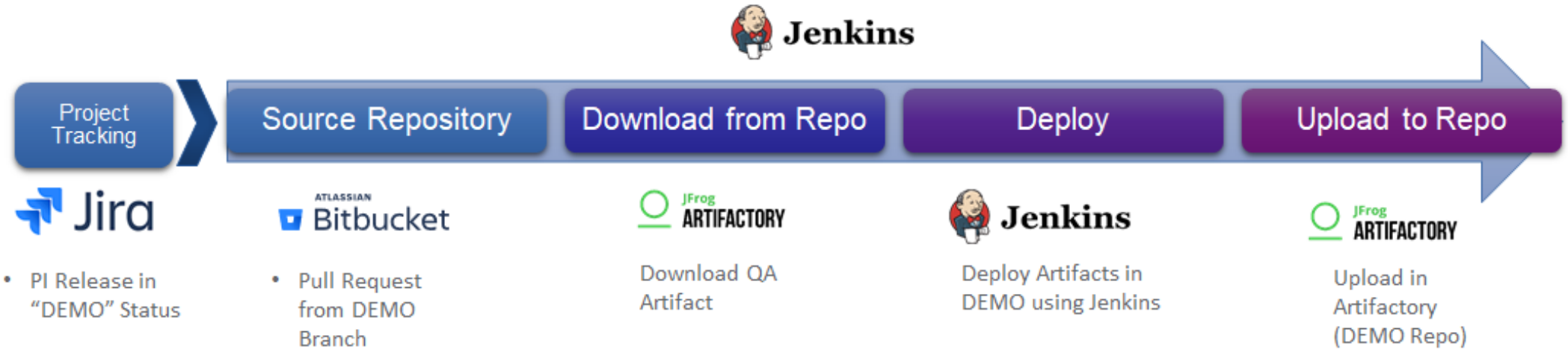
Best Practices and coverage scan

Vulnerabilities Scan

Upload in Artifactory using Maven

™

DEMO Env Pipeline



- PI Release in "DEMO" Status

- Pull Request from DEMO Branch

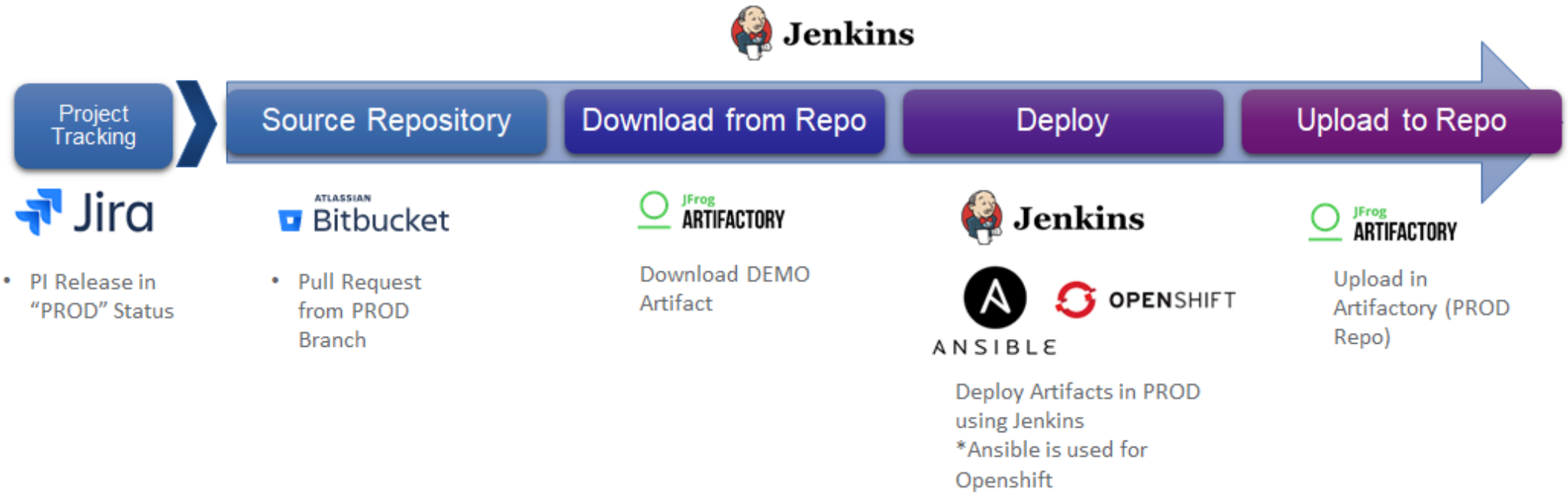
Download QA Artifact

Deploy Artifacts in DEMO using Jenkins

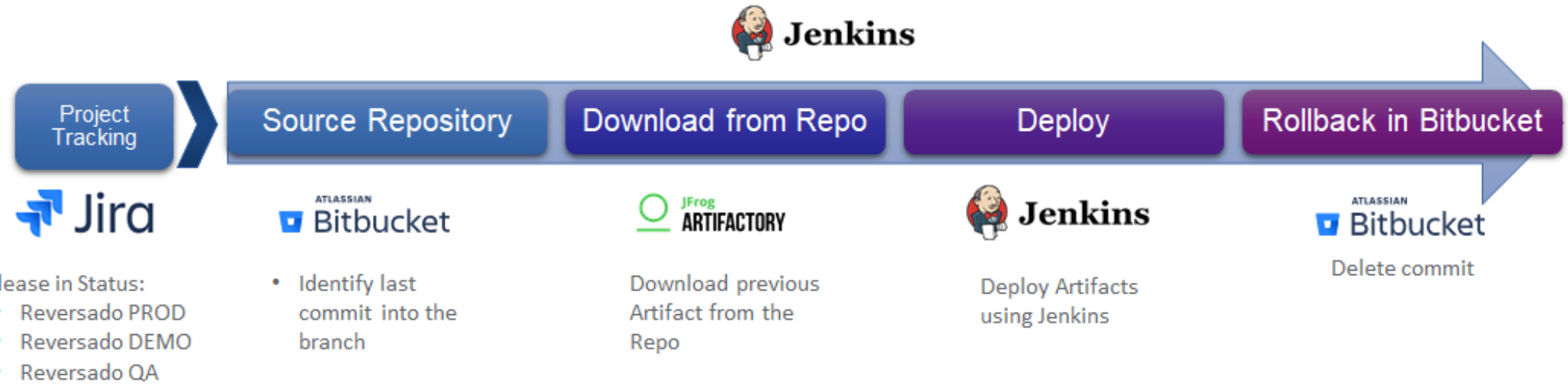
Upload in Artifactory (DEMO Repo)

erian™

PROD Env Pipeline

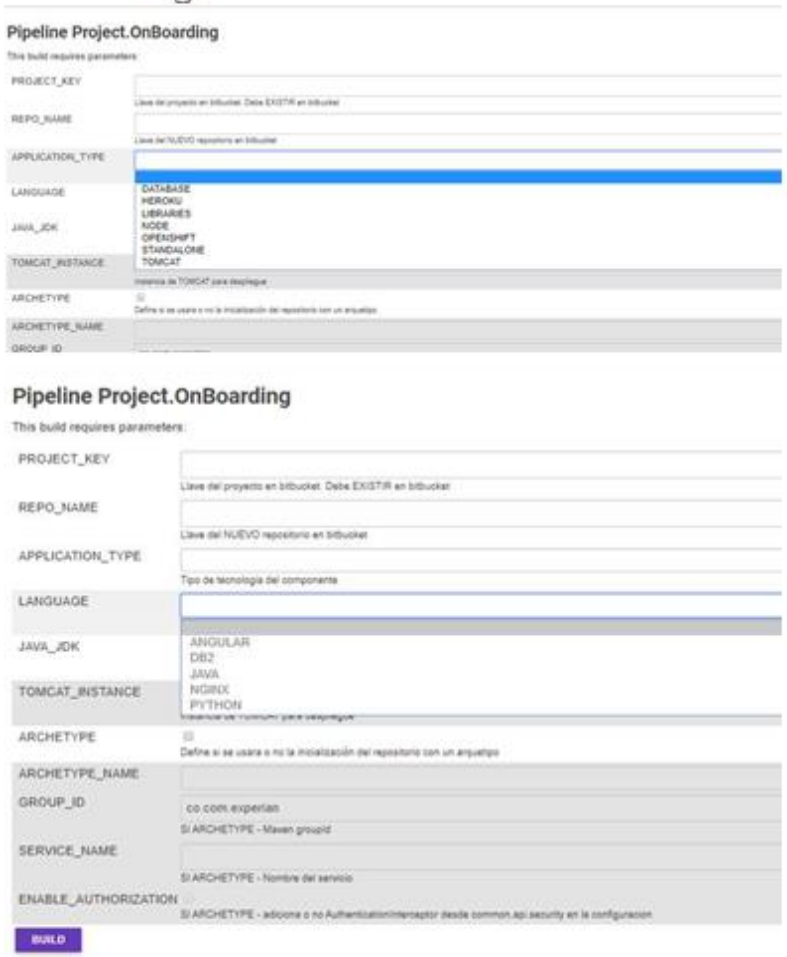


Rollback Pipeline



Onboarding Process

Jenkins



ATLASSIAN Bitbucket

DATABASE-DB2.jenkins.yml	python template
HEROKU-ANGULAR.jenkins.yml	python template
HEROKU-JAVA.jenkins.yml	python template
HEROKU-NGINX.Dockerfile	default docker file
HEROKU-NGINX.jenkins.yml	tmpl
LIBRARIES-COMMONS.jenkins.yml	libs order
LIBRARIES-JAVA.jenkins.yml	lib
NODE-ANGULAR.jenkins.yml	Change
OPENSHIFT-ANGULAR.jenkins.yml	sonar proj name
OPENSHIFT-JAVA.Dockerfile	openshift nginx
OPENSHIFT-JAVA.jenkins.yml	sonar afterbuild openshift java
OPENSHIFT-NGINX.Dockerfile	updates
OPENSHIFT-NGINX.jenkins.yml	default docker file
STANDALONE-JAVA.jenkins.yml	standalone deploy info
STANDALONE-PYTHON.jenkins.yml	standalone py
TOMCAT-ANGULAR-Deploy.conf	tomcat envs
TOMCAT-ANGULAR.jenkins.yml	TOMCAT ANGULAR
TOMCAT-JAVA-Deploy.conf	tomcat envs
TOMCAT-JAVA.jenkins.yml	tmpl fixes

ATLASSIAN Bitbucket

Repository with
4 Branches:

- DEV + base files(git ignore, template,)
- QA + base files
- DEMO
- PROD

Shared Libraries

ansibleCreateRoute	db2InitRemoteServer	jobCleanWs	standalone
ansibleMonitorRun	db2LogCatalog	jobEnd	veracodeDashboard
ansibleRun	db2SendChngSt	jobGetCommit	veracodeQueue
artifactoryCheckCommit	dockerBuild	jobGetConfig	veracodeScan
artifactoryDeployLibs	getInfoFromRepo	jobGetEnvType	zipFiles
artifactoryDwnld	getParsedJson	jobGetProjName	
artifactoryPromote	getSimpleRegex	jobInitSshServer	
artifactoryUpld	getUrlData	jobMainStart	
bitBucketCheckCommit	heroku	jobPrepareWS	
bitBucketCommId	herokuGetReleaseVersion	jobRun	
bitbucketCreatePullRequest	herokuGetTag	jobRunStages	
bitBucketGetChangeSet	herokuGetTagVersion	jobSetStages	
bitBucketGetPrevCommit	herokuPromote	jobSetStatus	
bitbucketIsMergeCommit	herokuRollBack	jobSetupEnv	
bitBucketRollback	herokuTagVer	jobShareLog	
bitBucketSearchIssueBranchs	herokuValidateRll	jobStage	
bitbucketSetBranches	jiraComm	jobValidateRllBck	
bitbucketSolveAllPR	jiraCommitsInfo	mapMerge	
bitbucketSolveMerge	jiraCommitsRllBck	mvn	
bitBucketToRef	jiraDeployData	openshift	
bitBucketValidatePR	jiraGetAttr	openshiftCheckProject	
cmd	jiraGetTaskBranchInfo	openshiftCheckService	
console	jiraIssueEdit	openshiftContainer	
db2	jiraIssueInfo	openshiftGetDetailsApp	
db2CompareDdl	jiraValidateIssue	openshiftLogin	
db2ExecuteDB	jobAppDetails	openshiftLogout	
db2ExtractDdl	jobBuild	openshiftPackage	
	jobBuildTechs	sonarQubeScan	

Functionality to Leverage

Shared Libraries to analyze and migrate to PLaaS – 5 7 days estimated effort

- **Ansible**

ansibleCreateRoute
ansibleMonitorRun
ansibleRun

- **OpenShift**

openshift
openshiftCheckProject
openshiftCheckService
openshiftContainer
openshiftGetDetailsApp

- **Veracode**

veracodeQueue

- **Utils**

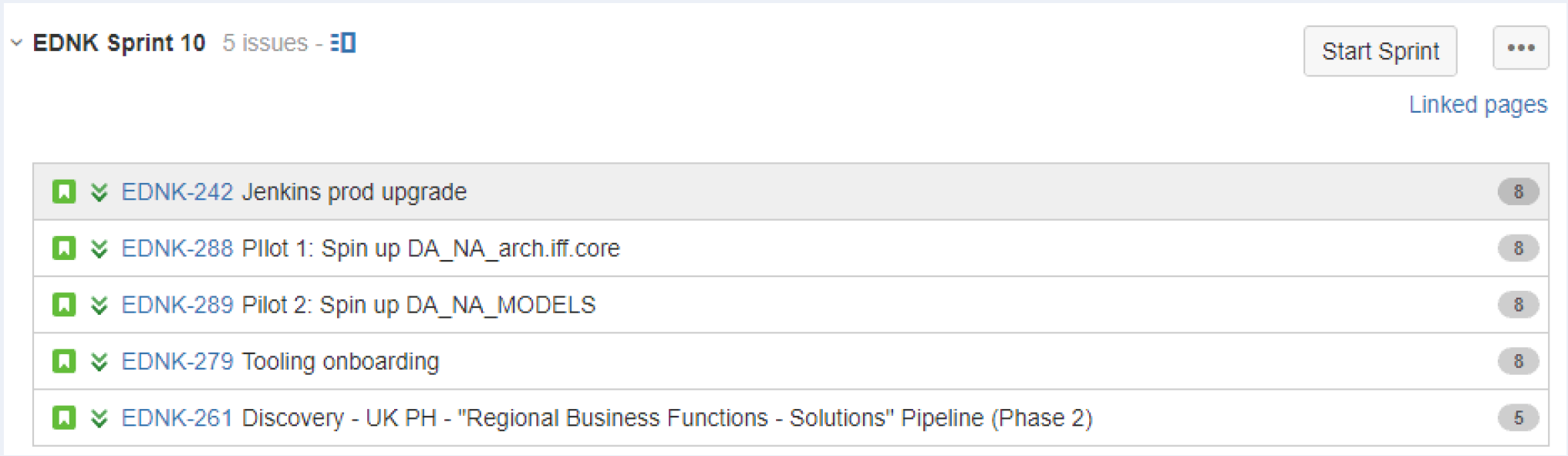
zipFiles
Tomcat
cmd.Groovy (for call NMP build instruction: cmd: npm set strict-ssl false && npm install --verbose -P -D -O)

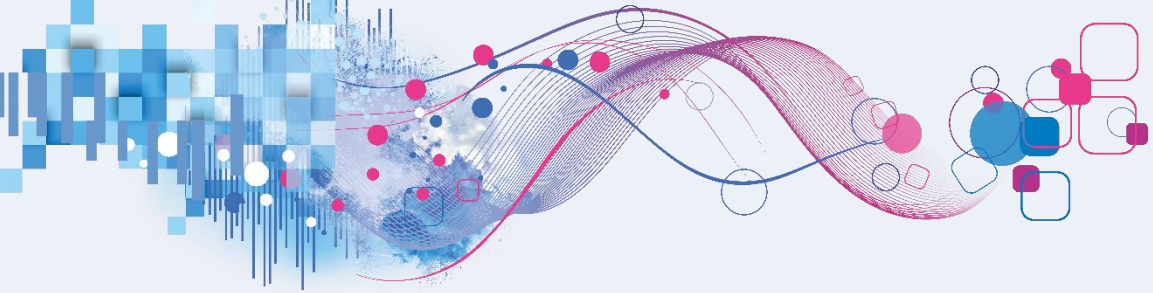
- **Docker**

dockerBuild
dockerPush
dockerTag
dockerLogin

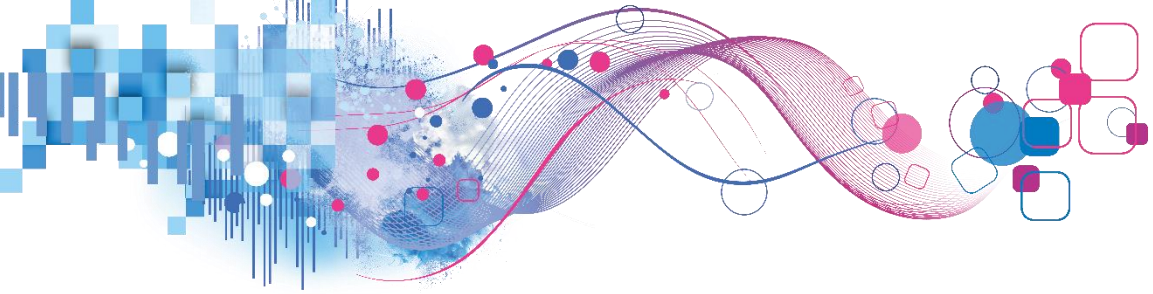
™

Next Steps


EDNK Sprint 10 5 issues -
Start Sprint
Linked pages
EDNK-242 Jenkins prod upgrade 8
EDNK-288 PIlot 1: Spin up DA_NA_arch.iff.core 8
EDNK-289 Pilot 2: Spin up DA_NA_MODELS 8
EDNK-279 Tooling onboarding 8
EDNK-261 Discovery - UK PH - "Regional Business Functions - Solutions" Pipeline (Phase 2) 5



Feedback / Questions?




experian
TM