



DevOps World



Jenkins World

**Make it scale:
Intelligent Jenkins Pipelines**



About us ...

Sven Merk



Oliver Nocon



About SAP

- 100% of the most successful companies in Portugal run SAP solutions.
Source: Forbes Global 2000
- Our customers produce more than 82% of the coffee and tea we drink each day.
- Every 60 seconds
Our customers produce 12,600 kilograms of chocolate.
- Our customers produce more than 77% of the world's beer.



Goals

Scale!

Maximize re-use (efficiency, productivity, ...)

Keep teams autonomous

Project “Piper”: Shared coded knowledge – ready to use ...

Jenkins Pipelines as Code

Piper ready-made pipelines

Jenkins Shared Libraries

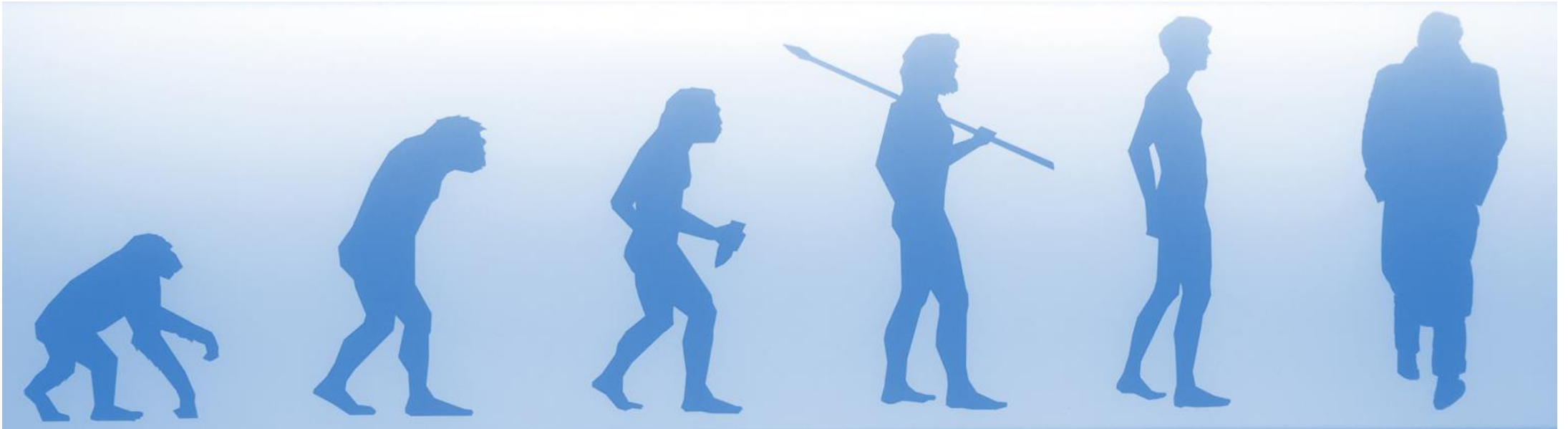
Piper Jenkins library

Docker Images

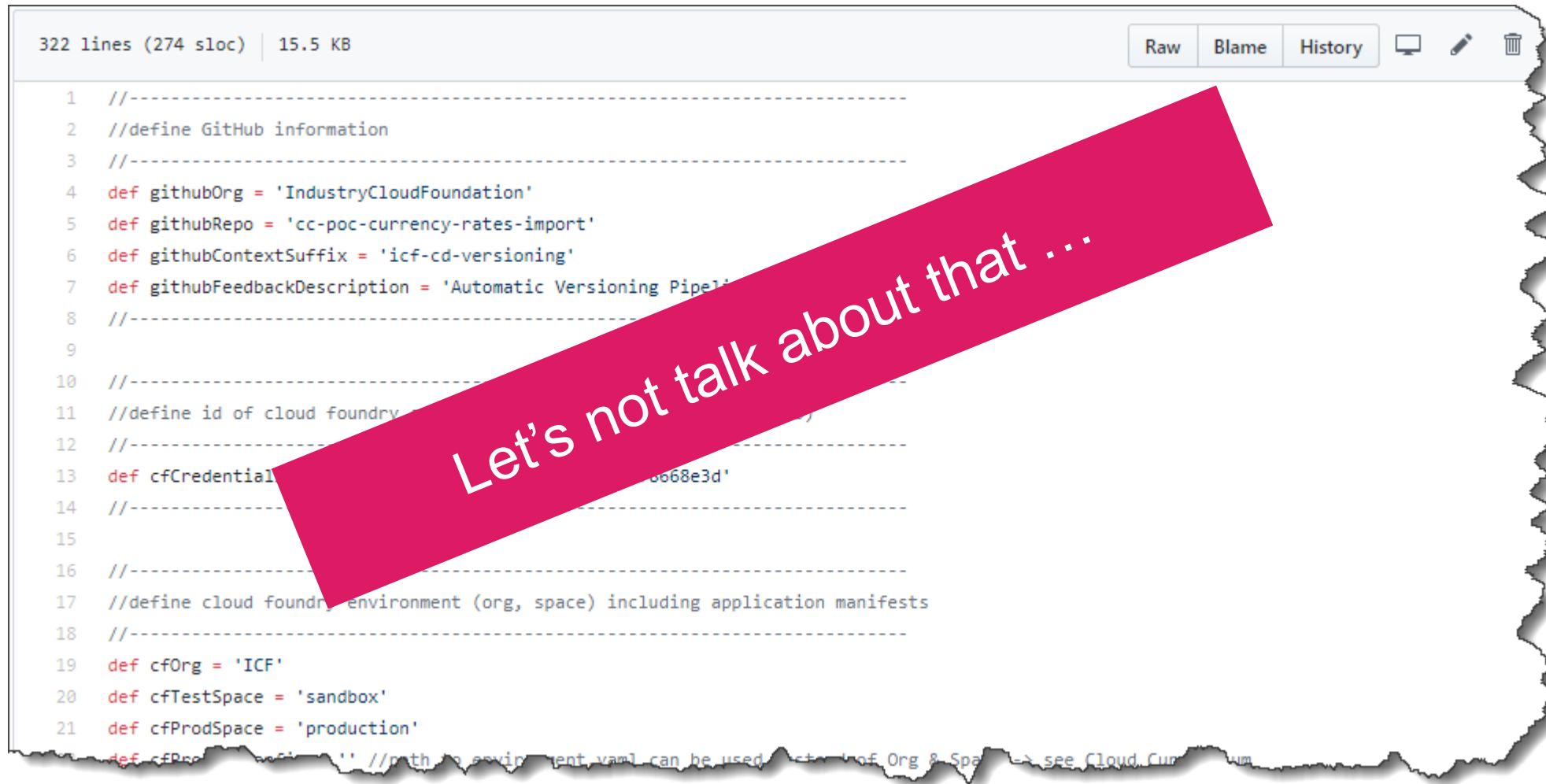
Piper images

Tailored Jenkins: `ppiper/jenkins-master` with `cx-server` toolkit

The journey ...



Jenkinsfile



322 lines (274 sloc) | 15.5 KB

Raw Blame History

```
1 //-----
2 //define GitHub information
3 //-----
4 def githubOrg = 'IndustryCloudFoundation'
5 def githubRepo = 'cc-poc-currency-rates-import'
6 def githubContextSuffix = 'icf-cd-versioning'
7 def githubFeedbackDescription = 'Automatic Versioning Pipeline'
8 //-----
9
10 //-----
11 //define id of cloud foundry
12 //-----
13 def cfCredential = 'cf-credentials-8668e3d'
14 //-----
15
16 //-----
17 //define cloud foundry environment (org, space) including application manifests
18 //-----
19 def cfOrg = 'ICF'
20 def cfTestSpace = 'sandbox'
21 def cfProdSpace = 'production'
22 def cfEnv = 'prod' //path to environment yaml can be used instead of Org & Space -> see Cloud Currency
```

Jenkinsfile + helper.groovy + config.properties



The image shows a screenshot of a Jenkinsfile with 476 lines (412 sloc) and a size of 25.2 KB. The file is displayed in a code editor with a light blue background. A yellow highlight is placed over a comment line: `//Only change code below if you know what your are doing`. Three pink callout boxes with white text are overlaid on the right side of the code. The first callout, pointing to the top of the code, says "Dev project within a dev project ...". The second callout, pointing to the middle of the code, says "Redundancies across projects". The third callout, pointing to the bottom of the code, says "Decoupling from innovations". The code itself starts with a shebang `#!/usr/bin/env groovy` and includes several imports. It then defines several URLs for helper scripts and Nexus milestones/releases. The code is partially obscured by the callouts and the yellow highlight.

```
476 lines (412 sloc) | 25.2 KB
1  #!/usr/bin/env groovy
2  //----- //-----
3  import
4  import
5  import
6  //Only change code below if you know what your are doing
7  //-----
8  //-----
9  //Only
10 //-----
11 //-----
12
13 def helperScriptUrl = 'https://github. /raw/ContinuousDelivery/jenkins-pipelines/master/scripts/helper.groovy'
14 def fortifyScriptUrl = 'https://github. /raw/ContinuousDelivery/jenkins-pipelines/master/scripts/fortifychecks.py'
15
16 def nexusMilestoneUrl = 'http://nexus. /nexus/content/repositories/deploy.milestones'
17 def nexusReleaseUrl = 'http://nexus. /nexus/content/repositories/deploy.releases'
18
19 @Field def helper
20 node{
21     delete('')
```

Dev project within a dev project ...

Redundancies across projects

Decoupling from innovations

Jenkinsfile + library + config.yml

```
200 lines (186 sloc) | 9 KB
Raw

1  #!/usr/bin/env groovy
2  @Library('piper-lib') _
3
4  try {
5      // pull request voting
6      if (env.BRANCH_NAME.startsWith('PR')) {
7          stage('Pull-request voting') {
8              node {
9                  deleteDir()
10                 checkout scm
11
12                 setupPipelineEnvironment script: this, storeGithubStatistics: false
13
14                 measureDuration(script: this, measurementName: 'voter_duration') {
15                     executeDocker(dockerImage: 'docker.io/robertnixon/icf/node', dockerOptions: '-v /var/run/docker.sock:/var/run/docker.sock') {
16                         sh "npm install && npm test"
17                     }
18                     publishCheckResults eslint: [pattern: '**/eslint.jslint.xml'], a
19                     publishTestResults junit: [archive: true, pattern: '**/TEST-*.xml']
20                 }
21             }
22         }
23     }
24 }
25 // master pipeline
```

Increased productivity
and ease of use

Dev project within a
dev project ...

Redundancies across
projects

Decoupling from
innovations

Ready-made pipeline + config.yml

```
4 lines (2 sloc) | 55 Bytes
1 @Library('piper-lib-os') _
2
3 piperPipeline script: this

37 lines (37 sloc) | 1.51 KB
1 general:
2   buildTool: docker
3   dockerCredentialsId: Artifactory
4   dockerImageName: piperstaging/golang
5   dockerRegistryUrl: https://docker.
6   githubApiUrl: https://github.
7   gitSshKeyCredentialsId: GitHub_Test_SSH
8   verbose: true
9 stages:
10  Build:
11    dockerImageName: piperstaging/golang
12  Promote:
13    containerPushToRegistry: true
14 steps:
15  artifactSetVersion:
```

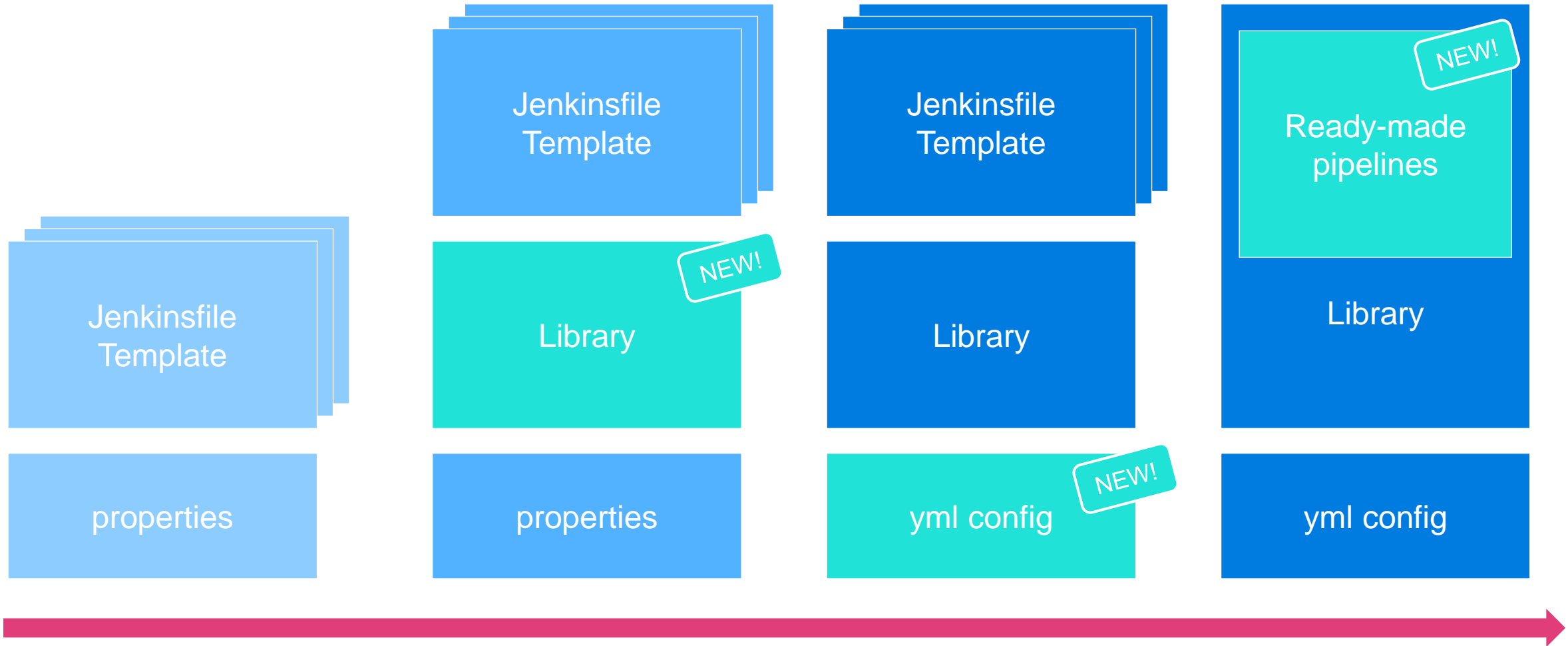
No-code adoption of Continuous Delivery

No investments into building, carrying the pipeline

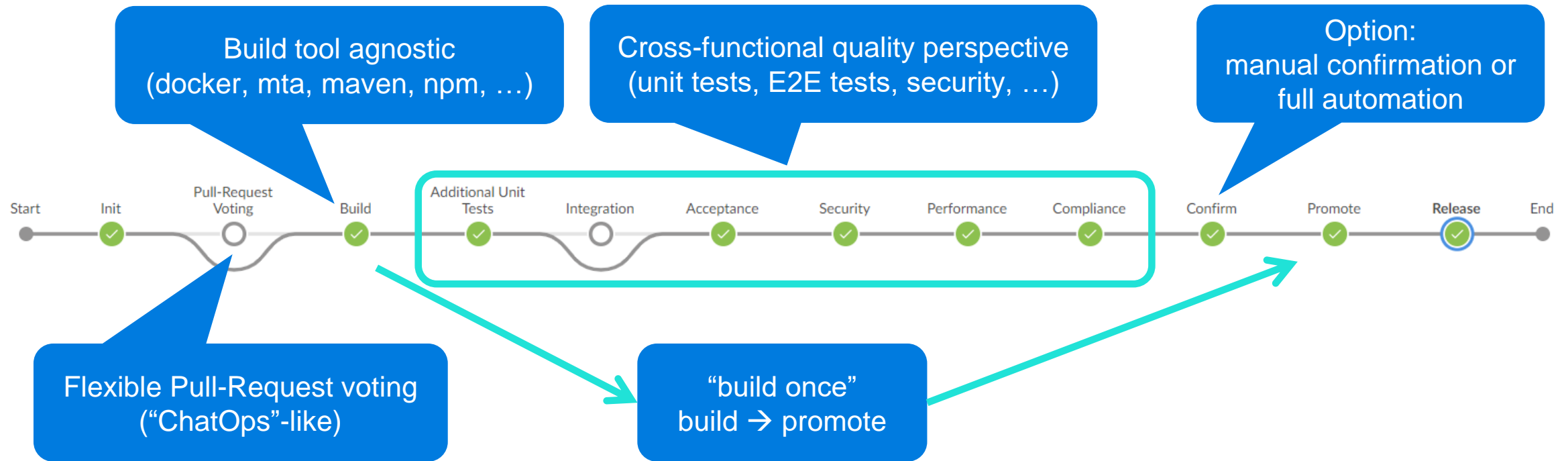
More consistency

Getting & operating Jenkins system might still be troubling

Let's recap: our maturity levels ...

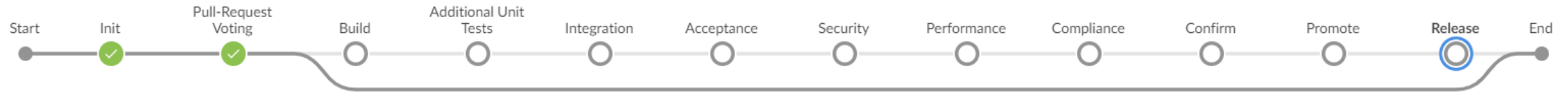


Project “Piper”: General purpose pipeline



More information: <https://sap.github.io/jenkins-library/stages/introduction/>

How you get started in a minute ...



Pipeline script (Jenkins)

```
@Library('piper-lib-os') _  
  
piperPipeline script: this
```

Initial config (Pull-Request voting, other things can be added iteratively)

```
general:  
  buildTool: npm
```

Separation of pipeline logic and configuration

Pipeline script (Jenkins)

```
@Library('piper-lib-os') _  
piperPipeline script: this
```

Provided

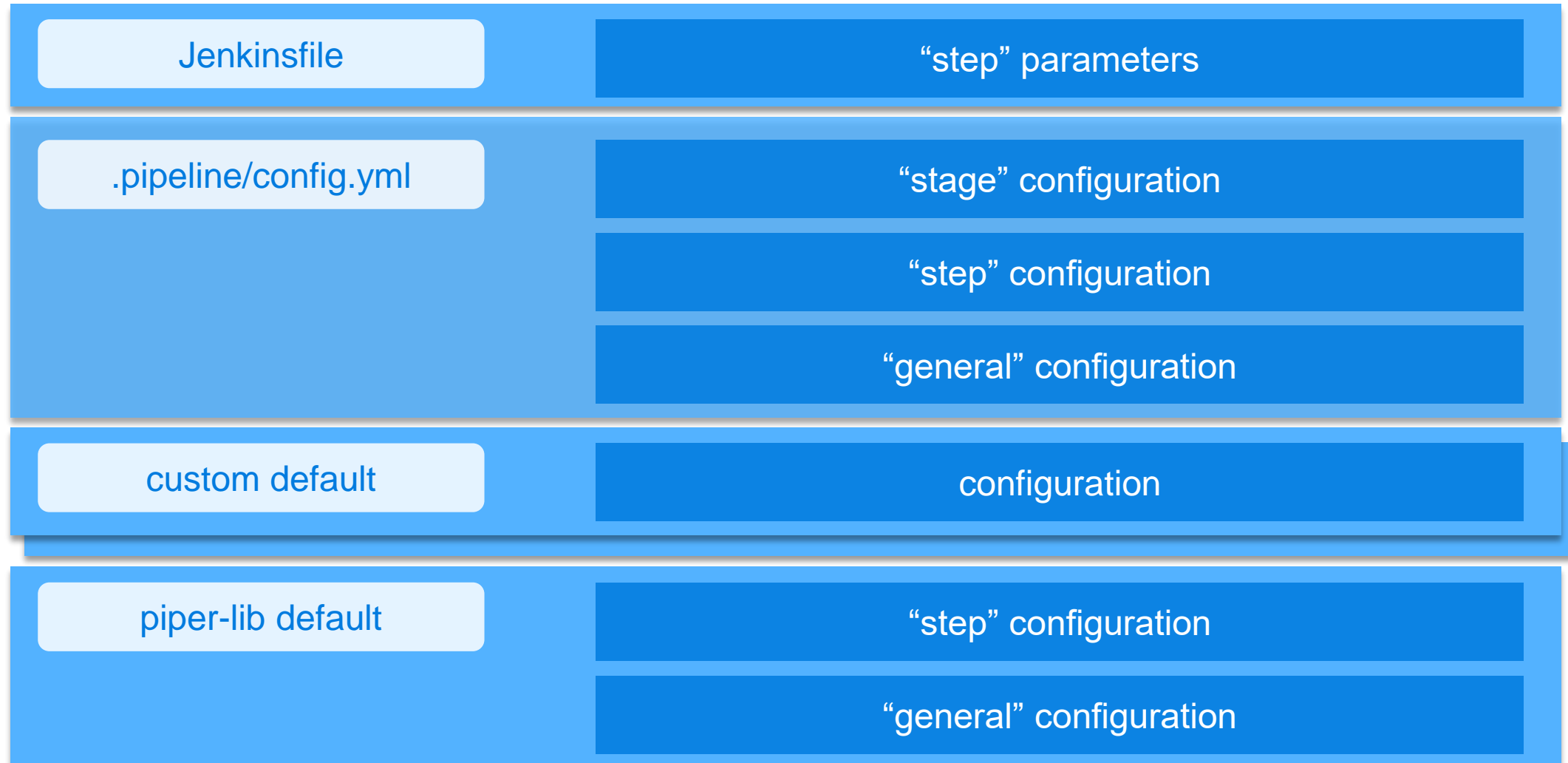
Configuration

```
general:  
  buildTool: npm  
stages:  
  Security:  
    verbose: true  
steps:  
  whitesourceExecuteScan:  
    productName: JenkinsWorld  
  ...
```

Stage extension

Team-owned

Configuration - layered



Smart Pipeline logic ...

```
1  stages:
2    Init:
3      stepConditions:
4        slackSendNotification:
5          configKeys:
6            - 'channel'
7      'Pull-Request Voting': {}
8    Build: {}
9    'Additional Unit Tests':
10     stepConditions:
11       batsExecuteTests:
12         filePattern: '**/*.bats'
13       karmaExecuteTests:
14         filePattern: '**/karma.conf.js'
15    Integration: {}
16    Acceptance:
17     stepConditions:
18       cloudFoundryDeploy:
```

comes with the pipeline definition
[https://github.com/SAP/jenkins-](https://github.com/SAP/jenkins-library/blob/master/resources/com.sap.piper/pipeline/stageDefaults.yml)
[library/blob/master/resources/com.sap.piper/pipeline/stageDefaults.yml](https://github.com/SAP/jenkins-library/blob/master/resources/com.sap.piper/pipeline/stageDefaults.yml)

Extensibility

- 1

Stage exit

`.pipeline/extensions/<StageName>.groovy`
- 2

Central custom template

Custom Library Step (similar to `piperPipeline`)
- 3

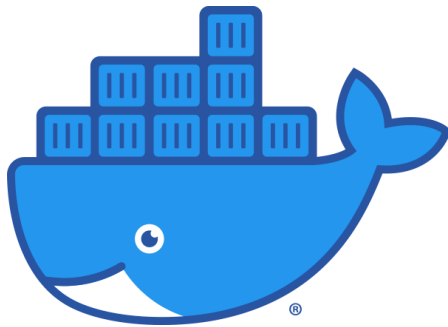
De-central custom template

Custom Jenkinsfile

Transparent Docker execution

Pipeline script or pipeline step

```
...  
dockerExecute(dockerImage: 'node:latest') {  
    sh "npm install"  
}  
...
```



Shift security left ...

The screenshot shows a GitHub pull request titled "Update go-piper.go #4". The interface includes a sidebar on the left with a conversation list, a main area with commit details, and a right sidebar with check status. A green box highlights the "Merge pull request" button in the right sidebar, which is accompanied by the text "You can also open this in GitHub Desktop or view command line instructions." A yellow box highlights the "Merge pull request" button in the main area, also with the same text. A blue box highlights the comment input field at the bottom, containing the text "/piper whitesource".

Update go-piper.go #4

Open [user] wants to merge 1 commit into ContinuousDelivery/go-piper.

Conversation 7

No description provided

Update go-piper.go

Some checks have passed

1 pending check

This pull request has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

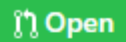
You can also open this in GitHub Desktop or view command line instructions.

commented 12 minutes ago

/piper whitesource

Shift security left ...

Update go-piper.go #4



wants to merge 1 commit into

Add more commits by pushing to the `test-prVoting` branch on `ContinuousDelivery/go-piper`.

WhiteSource Security Vulnerability Report

WhiteSource product name: SHC - Piper

Filtered project names: [github. /ContinuousDelivery/go-piper - PR-4]

total number of vulnerabilities: 0

total number of high/critical vulnerabilities with CVSS score ≥ 7 : 0

Snapshot taken: Nov 18, 2019 - 11:23:21 UTC

Entry #	Date	CVE	CVSS Score	CVSS Version	Project	Library file name	Library group ID	Library artifact ID	Library version	Description	Top fix
---------	------	-----	------------	--------------	---------	-------------------	------------------	---------------------	-----------------	-------------	---------

No publicly known vulnerabilities detected



commented 12 minutes ago

Author

Member



/piper whitesource

Quality process – behind the scenes ...

```
1 import org.junit.Before
2 import org.junit.Rule
3 import org.junit.Test
4 import org.junit.rules.ExpectedException
5 import org.junit.rules.RuleChain
6 import util.*
7
8 import static org.hamcrest.Matchers.*
9 import static org.junit.Assert.assertThat
10
11 class KarmaExecuteTestsTest extends BasePiperTest {
12     private JenkinsStepRule stepRule = new JenkinsStepRule()
13     private JenkinsLoggingRule loggingRule = new JenkinsLoggingRule()
14     private JenkinsShellCallRule shellRule = new JenkinsShellCallRule()
15     private JenkinsEnvironmentRule environmentRule = new JenkinsEnvironmentRule()
16     private ExpectedException thrown = ExpectedException.none()
17
18     @Rule
19     public RuleChain rules = Rules
20         .getCommonRules(this)
21         .around(new JenkinsReadYamlRule(this))
22         .around(shellRule)
23         .around(loggingRule)
24         .around(environmentRule)
25         .around(stepRule)
26         .around(thrown)
27
28     def seleniumParams = [:]
```

```
40 @Test
41 void testDefaults() throws Exception {
42     stepRule.step.karmaExecuteTests(
43         script: nullScript,
44         juStabUtils: utils
45     )
46     assertThat(shellRule.shell, hasItems(
47         containsString("cd '.' && npm install --quiet"),
48         containsString("cd '.' && npm run karma")
49     ))
50     assertThat(seleniumParams.dockerImage, is('node:8-stretch'))
51     assertThat(seleniumParams.dockerName, is('karma'))
52     assertThat(seleniumParams.dockerWorkspace, is('/home/node'))
53     assertJobStatusSuccess()
54 }
55
```

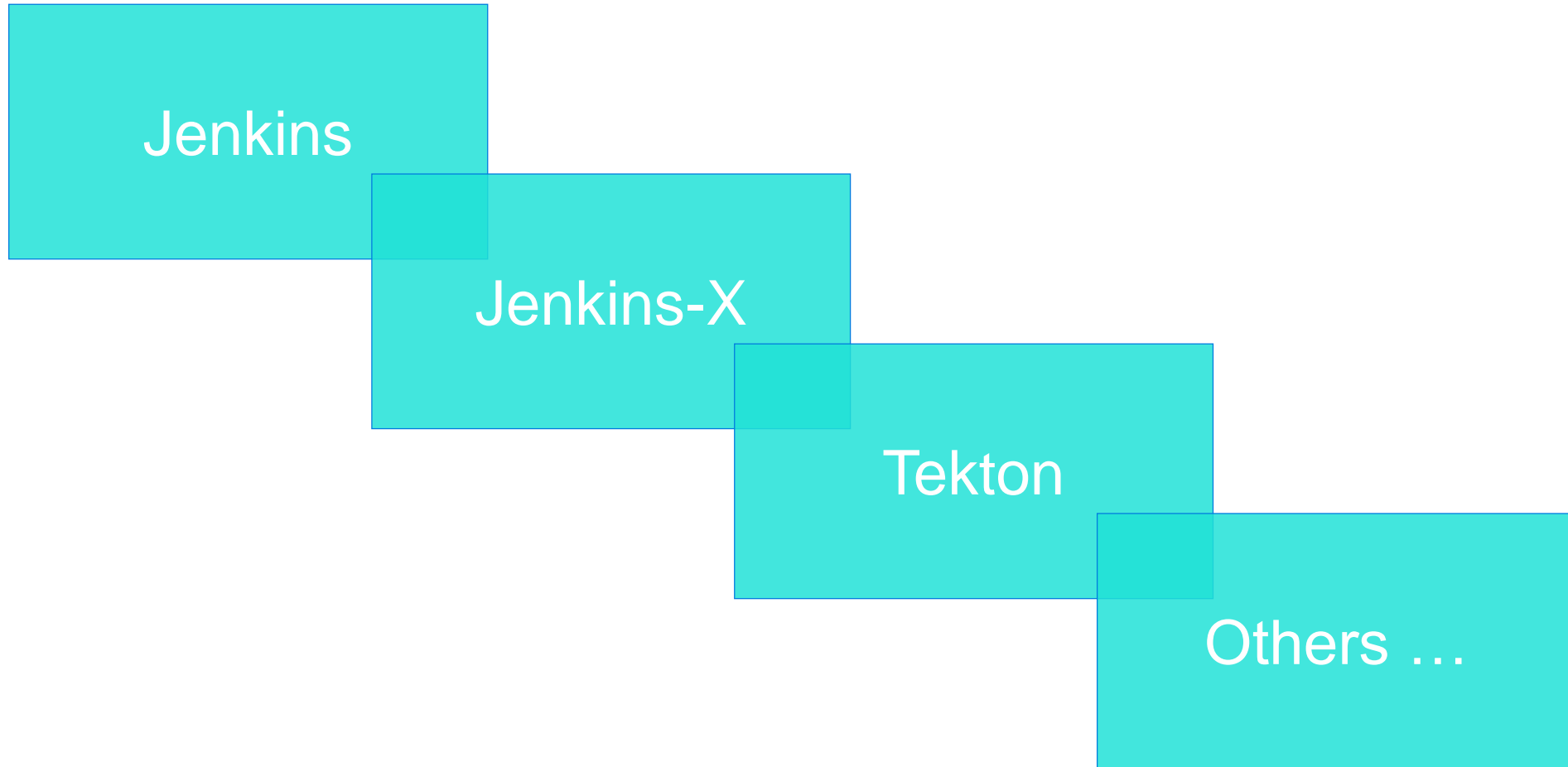
JenkinsPipelineUnit
Spring Test Framework

763 tests in
~1.5 minutes



CD.FOUNDATION

Environment is evolving ...



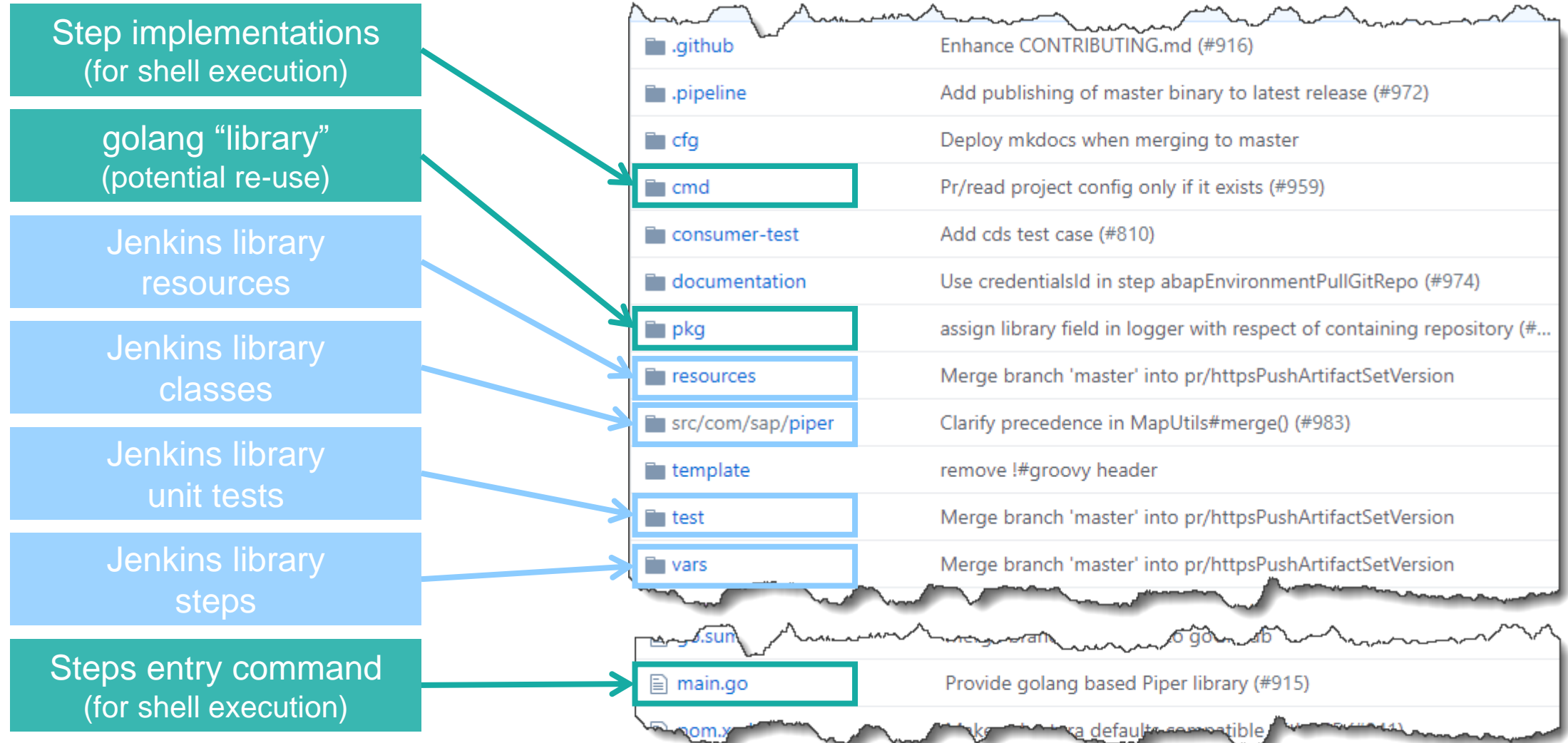
Jenkins library step today (config resolution)

```
69 @GenerateDocumentation
70 void call(Map parameters = [:]) {
71     handlePipelineStepErrors(stepName: STEP_NAME, stepParameters: parameters) {
72         def script = checkScript(this, parameters) ?: this
73
74         // load default & individual configuration
75         Map config = ConfigurationHelper.newInstance(this)
76             .loadStepDefaults()
77             .mixinGeneralConfig(script.commonPipelineEnvironment, GENERAL_CONFIG_KEYS)
78             .mixinStepConfig(script.commonPipelineEnvironment, STEP_CONFIG_KEYS)
79             .mixinStageConfig(script.commonPipelineEnvironment, parameters.stageName?:env.STAGE_NAME, STEP_CONFIG_KEYS)
80             .mixin(parameters, PARAMETER_KEYS)
81             .addIfEmpty('githubOrg', script.commonPipelineEnvironment.getGithubOrg())
82             .addIfEmpty('githubRepo', script.commonPipelineEnvironment.getGithubRepo())
83             .addIfEmpty('version', script.commonPipelineEnvironment.getArtifactVersion())
84             .withMandatoryProperty('githubOrg')
85             .withMandatoryProperty('githubRepo')
86             .withMandatoryProperty('githubTokenCredentialsId')
87             .withMandatoryProperty('version')
88             .use()
```

Jenkins library step today (execution)...

```
93
94     def releaseBodyHeader = ''
95     if (config.releaseBodyHeader) {
96         releaseBodyHeader = GStringTemplateEngine.newInstance()
97             .createTemplate(config.releaseBodyHeader)
98             .make([
99                 config: config,
100                 commonPipelineEnvironment: script.commonPipelineEnvironment
101             ]).toString()
102         releaseBodyHeader += '<br />'
103     }
104     def releaseBody = releaseBodyHeader
105     def content = getLastRelease(config, TOKEN)
106     if (config.addClosedIssues)
107         releaseBody += addClosedIssue(config, TOKEN, content.published_at)
108     if (config.addDeltaToLastRelease)
109         releaseBody += addDeltaToLastRelease(config, content.tag_name)
110     postNewRelease(config, TOKEN, releaseBody)
111
```

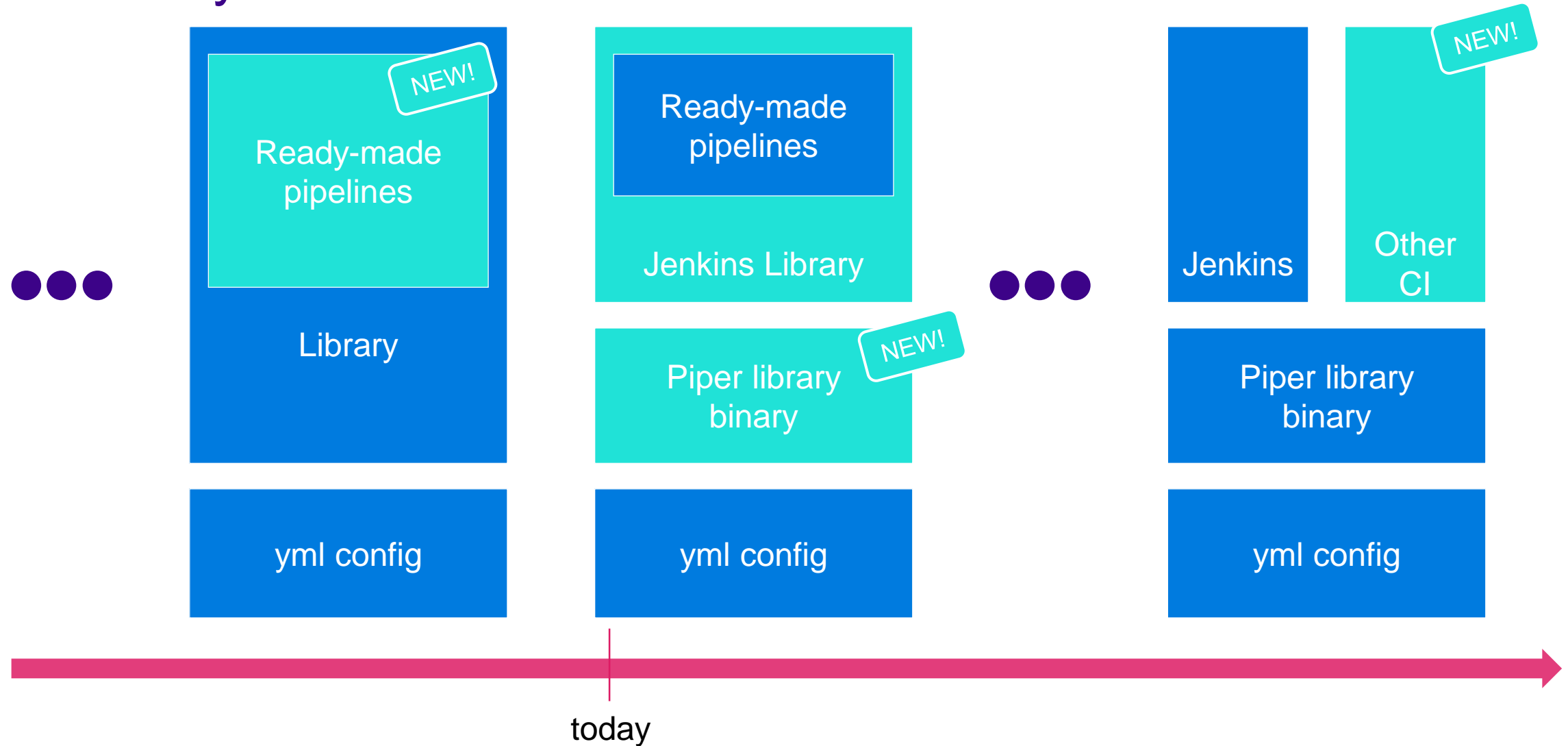
New library structure



New Jenkins library steps

```
23     new GroovyLib(this, utils).unstashPiperBin()
24     utils.unstash('pipelineConfigAndTests')
25
26     writeFile(file: METADATA_FILE, text: libraryResource(METADATA_FILE))
27
28     withEnv([
29         "PIPER_parametersJSON=${groovy.json.JsonOutput.toJson(parameters)}",
30         "PIPER_owner=${commonPipelineEnvironment.getGithubOrg() ?: ''}",
31         "PIPER_repository=${commonPipelineEnvironment.getGithubRepo() ?: ''}",
32         "PIPER_version=${commonPipelineEnvironment.getArtifactVersion() ?: ''}"
33     ]) {
34         // get context configuration
35         config = readJSON (text: sh(returnStdout: true, script: "./piper getConfig --contextConfig --stepMetadata '${METADATA_FILE}' --"))
36
37         // execute step
38         withCredentials([string(credentialsId: config.githubTokenCredentialsId, variable: 'TOKEN')]) {
39             sh "./piper githubPublishRelease --token ${TOKEN}"
40         }
41     }
```

The story does not end here ...



Scale BIG with Jenkins Pipelines

Gain efficiency & productivity with
“ready-made pipelines”

Autonomy of teams strengthened



<https://sap.github.io/jenkins-library/>

Project "Piper": Continuous Delivery for the SAP Ecosystem

Home

Guided Tour

Configuration

Pipelines ▾

Scenarios ▾

Extensibility

Library steps ▾

Resources ▾

Project "Piper" User Documentation

Continuous delivery is a method to develop software with short feedback cycles. It is applicable to projects both for SAP Cloud Platform and SAP on-premise platforms. SAP implements tooling for continuous delivery in project "Piper". The goal of project "Piper" is to substantially ease setting up continuous delivery in your project using SAP technologies.

What you get

To get you started quickly, project "Piper" offers you the following artifacts:

- A set of ready-made Continuous Delivery pipelines for direct use in your project
 - [General Purpose Pipeline](#)
 - [SAP Cloud SDK Pipeline](#)
- A [shared library](#) that contains reusable step implementations, which enable you to customize our preconfigured pipelines, or to even build your own customized ones
- A set of [Docker images](#) to setup a CI/CD environment in minutes using sophisticated life-cycle management

Table of contents

The do-it-yourself way,
with Library

Extensibility

API

DevOps World



Jenkins World

Thank You!

