```
#!/usr/bin/env groovy

/*
 * This Jenkinsfile is intended to run on https://ci.jenkins.io and may
fail anywhere else.
 * It makes assumptions about plugins being installed, labels mapping to
nodes that can build what is needed, etc.
 */

def failFast = false

properties([
  buildDiscarder(logRotator(numToKeepStr: '50', artifactNumToKeepStr:
'3')),
  disableConcurrentBuilds(abortPrevious: true)
])

def axes = [
  platforms: ['linux', 'windows'],
  jdks: [11, 17, 21],
]

stage('Record build') {
  retry(conditions: [kubernetesAgent(handleNonKubernetes: true),
nonresumable()], count: 2) {
    node('maven-11') {
      infra.checkoutSCM()

      /*
       * Record the primary build for this CI job.
       */
      withCredentials([string(credentialsId: 'launchable-jenkins-
jenkins', variable: 'LAUNCHABLE_TOKEN')]) {
        /*
         * TODO Add the commits of the transitive closure of the Jenkins
WAR under test to this build.
         */
        sh 'launchable verify && launchable record build --name
${BUILD_TAG} --source jenkinsci/jenkins=.'
        axes.values().combinations {
          def (platform, jdk) = it
          if (platform == 'windows' && jdk != 17) {
            return // unnecessary use of hardware
          }
          def sessionFile = "launchable-session-${platform}-
jdk${jdk}.txt"
          sh "launchable record session --build ${env.BUILD_TAG} --flavor
platform=${platform} --flavor jdk=${jdk} >${sessionFile}"
          stash name: sessionFile, includes: sessionFile
        }
      }

      /*
```

```
       * Record commits for use in downstream CI jobs that may consume
this artifact.
       */
      withCredentials([string(credentialsId: 'launchable-jenkins-
acceptance-test-harness', variable: 'LAUNCHABLE_TOKEN')]) {
        sh 'launchable verify && launchable record commit'
      }
      withCredentials([string(credentialsId: 'launchable-jenkins-bom',
variable: 'LAUNCHABLE_TOKEN')]) {
        sh 'launchable verify && launchable record commit'
      }
    }
  }
}

def builds = [:]

axes.values().combinations {
  def (platform, jdk) = it
  if (platform == 'windows' && jdk != 17) {
    return // unnecessary use of hardware
  }
  builds["${platform}-jdk${jdk}"] = {
    // see https://github.com/jenkins-
infra/documentation/blob/master/ci.adoc#node-labels for information on
what node types are available
    def agentContainerLabel = 'maven-' + jdk
    if (platform == 'windows') {
      agentContainerLabel += '-windows'
    }
    retry(conditions: [kubernetesAgent(handleNonKubernetes: true),
nonresumable()], count: 2) {
      node(agentContainerLabel) {
        // First stage is actually checking out the source. Since we're
using Multibranch
        // currently, we can use "checkout scm".
        stage("${platform.capitalize()} - JDK ${jdk} - Checkout") {
          infra.checkoutSCM()
        }

        def tmpDir = pwd(tmp: true)
        def changelistF = "${tmpDir}/changelist"
        def m2repo = "${tmpDir}/m2repo"
        def session

        // Now run the actual build.
        stage("${platform.capitalize()} - JDK ${jdk} - Build / Test") {
          timeout(time: 6, unit: 'HOURS') {
            dir(tmpDir) {
              def sessionFile = "launchable-session-${platform}-
jdk${jdk}.txt"
              unstash sessionFile
              session = readFile(sessionFile).trim()
            }
```

```
            def mavenOptions = [
              '-Pdebug',
              '-Penable-jacoco',
              '--update-snapshots',
              "-Dmaven.repo.local=$m2repo",
              '-Dmaven.test.failure.ignore',
              '-DforkCount=2',
              '-Dspotbugs.failOnError=false',
              '-Dcheckstyle.failOnViolation=false',
              '-Dset.changelist',
              'help:evaluate',
              '-Dexpression=changelist',
              "-Doutput=$changelistF",
              'clean',
              'install',
            ]
            if (env.CHANGE_ID && !pullRequest.labels.contains('full-
test')) {
              def excludesFile
              def target = platform == 'windows' ? '30%' : '100%'
              withCredentials([string(credentialsId: 'launchable-jenkins-
jenkins', variable: 'LAUNCHABLE_TOKEN')]) {
                if (isUnix()) {
                  excludesFile = "${tmpDir}/excludes.txt"
                  sh "launchable verify && launchable subset --session
${session} --target ${target} --get-tests-from-previous-sessions --
output-exclusion-rules maven >${excludesFile}"
                } else {
                  excludesFile = "${tmpDir}\\excludes.txt"
                  bat "launchable verify && launchable subset --session
${session} --target ${target}% --get-tests-from-previous-sessions --
output-exclusion-rules maven >${excludesFile}"
                }
              }
              mavenOptions.add(0, "-
Dsurefire.excludesFile=${excludesFile}")
            }
            realtimeJUnit(healthScaleFactor: 20.0, testResults:
'*/target/surefire-reports/*.xml') {
              infra.runMaven(mavenOptions, jdk)
              if (isUnix()) {
                sh 'git add . && git diff --exit-code HEAD'
              }
            }
          }
        }

        // Once we've built, archive the artifacts and the test results.
        stage("${platform.capitalize()} - JDK ${jdk} - Publish") {
          archiveArtifacts allowEmptyArchive: true, artifacts:
'**/target/surefire-reports/*.dumpstream'
          // cli and war have been migrated to JUnit 5
          if (failFast && currentBuild.result == 'UNSTABLE') {
            error 'There were test failures; halting early'
```

```
          }
          if (platform == 'linux' && jdk == axes['jdks'][0]) {
            def folders = env.JOB_NAME.split('/')
            if (folders.length > 1) {
              discoverGitReferenceBuild(scm: folders[1])
            }
            recordCoverage(tools: [[parser: 'JACOCO', pattern:
'coverage/target/site/jacoco-aggregate/jacoco.xml']],
            sourceCodeRetention: 'MODIFIED', sourceDirectories: [[path:
'core/src/main/java']])

            echo "Recording static analysis results for
'${platform.capitalize()}'"
            recordIssues(
                enabledForFailure: true,
                tools: [java()],
                filters: [excludeFile('.*Assert.java')],
                sourceCodeEncoding: 'UTF-8',
                skipBlames: true,
                trendChartType: 'TOOLS_ONLY'
                )
            recordIssues(
                enabledForFailure: true,
                tools: [javaDoc()],
                filters: [excludeFile('.*Assert.java')],
                sourceCodeEncoding: 'UTF-8',
                skipBlames: true,
                trendChartType: 'TOOLS_ONLY',
                qualityGates: [[threshold: 1, type: 'TOTAL', unstable:
true]]
            )
            recordIssues([tool: spotBugs(pattern:
'**/target/spotbugsXml.xml'),
              sourceCodeEncoding: 'UTF-8',
              skipBlames: true,
              trendChartType: 'TOOLS_ONLY',
              qualityGates: [[threshold: 1, type: 'NEW', unstable:
true]]])
            recordIssues([tool: checkStyle(pattern:
'**/target/checkstyle-result.xml'),
              sourceCodeEncoding: 'UTF-8',
              skipBlames: true,
              trendChartType: 'TOOLS_ONLY',
              qualityGates: [[threshold: 1, type: 'TOTAL', unstable:
true]]])
            recordIssues([tool: esLint(pattern: '**/target/eslint-
warnings.xml'),
              sourceCodeEncoding: 'UTF-8',
              skipBlames: true,
              trendChartType: 'TOOLS_ONLY',
              qualityGates: [[threshold: 1, type: 'TOTAL', unstable:
true]]])
            recordIssues([tool: styleLint(pattern: '**/target/stylelint-
warnings.xml'),
```

```
              sourceCodeEncoding: 'UTF-8',
              skipBlames: true,
              trendChartType: 'TOOLS_ONLY',
              qualityGates: [[threshold: 1, type: 'TOTAL', unstable:
true]]])
            if (failFast && currentBuild.result == 'UNSTABLE') {
              error 'Static analysis quality gates not passed; halting
early'
            }
            def changelist = readFile(changelistF)
            dir(m2repo) {
              archiveArtifacts(
                  artifacts: "**/*$changelist/*$changelist*",
                  excludes: '**/*.lastUpdated,**/jenkins-
coverage*/,**/jenkins-test*/',
                  allowEmptyArchive: true, // in case we forgot to
reincrementalify
                  fingerprint: true
                  )
            }
          }
          withCredentials([string(credentialsId: 'launchable-jenkins-
jenkins', variable: 'LAUNCHABLE_TOKEN')]) {
            if (isUnix()) {
              sh "launchable verify && launchable record tests --session
${session} --flavor platform=${platform} --flavor jdk=${jdk} maven
'./**/target/surefire-reports'"
            } else {
              bat "launchable verify && launchable record tests --session
${session} --flavor platform=${platform} --flavor jdk=${jdk} maven
./**/target/surefire-reports"
            }
          }
        }
      }
    }
  }
}

def athAxes = [
  platforms: ['linux'],
  jdks: [17],
  browsers: ['firefox'],
]
athAxes.values().combinations {
  def (platform, jdk, browser) = it
  builds["ath-${platform}-jdk${jdk}-${browser}"] = {
    retry(conditions: [agent(), nonresumable()], count: 2) {
      node('docker-highmem') {
        // Just to be safe
        deleteDir()
        checkout scm
        infra.withArtifactCachingProxy {
          sh "bash ath.sh ${jdk} ${browser}"
```

```
        }
        junit testResults: 'target/ath-reports/TEST-*.xml',
testDataPublishers: [[$class: 'AttachmentPublisher']]
        /*
         * Currently disabled, as the fact that this is a manually
created subset will confuse Launchable,
         * which expects this to be a full build. When we implement
subsetting, this can be re-enabled using
         * Launchable's subset rather than our own.
         */
        /*
         withCredentials([string(credentialsId: 'launchable-jenkins-
acceptance-test-harness', variable: 'LAUNCHABLE_TOKEN')]) {
         sh "launchable verify && launchable record tests --no-build --
flavor platform=${platform} --flavor jdk=${jdk} --flavor
browser=${browser} maven './target/ath-reports'"
         }
         */
      }
    }
  }
}

builds.failFast = failFast
parallel builds
infra.maybePublishIncrementals()
```