# ESTA Tekton Parameters - estaTektonPipeline.json

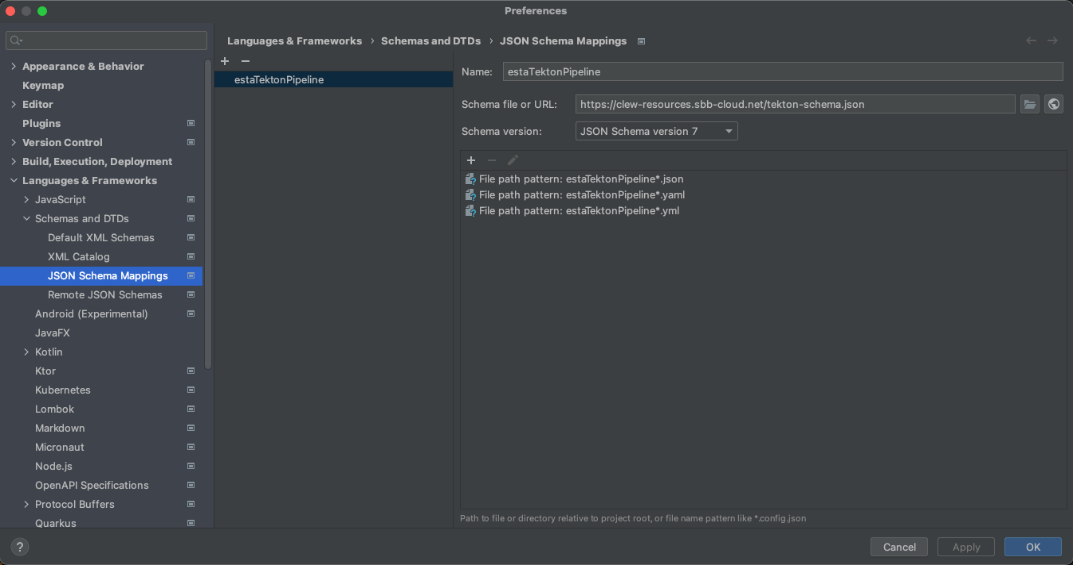
* [Pipelines](#ESTATektonParametersestaTektonPipeline.)
  + [Trigger criteria](#ESTATektonParametersestaTektonPipeline.)
  + [Build parameters](#ESTATektonParametersestaTektonPipeline.)
* [Stages](#ESTATektonParametersestaTektonPipeline.)
* [Helm Charts](#ESTATektonParametersestaTektonPipeline.)
* [Sonar Analysis](#ESTATektonParametersestaTektonPipeline.)
* [OWASP Dependency Check](#ESTATektonParametersestaTektonPipeline.)
* [Docker Settings](#ESTATektonParametersestaTektonPipeline.)
* [Examples](#ESTATektonParametersestaTektonPipeline.)
  + [Java (Maven)](#ESTATektonParametersestaTektonPipeline.)
  + [Node.js / NPM](#ESTATektonParametersestaTektonPipeline.)
  + [Golang](#ESTATektonParametersestaTektonPipeline.)
  + [Docker](#ESTATektonParametersestaTektonPipeline.)
  + [Parametrized Builds](#ESTATektonParametersestaTektonPipeline.)
  + [Custom Pipelines](#ESTATektonParametersestaTektonPipeline.)
* [Parameter Documentation](#ESTATektonParametersestaTektonPipeline.)
  + [Global Properties](#ESTATektonParametersestaTektonPipeline.)
  + [Docker configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Helm configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Builder configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Maven configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [NPM configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Flutter configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Go configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Python configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Parameters configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Buildpack configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Build pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Tagging pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Custom pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Sonar configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [GitGuardian secrets scan configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [OWASP dependency check configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Testcontainers configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Testreport configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Custom task configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Stage configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [ArgoCD configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Notification configuration structure](#ESTATektonParametersestaTektonPipeline.)
  + [Advanced Build Settings configuration structure](#ESTATektonParametersestaTektonPipeline.)
* [Pre-defined Pipeline Parameters](#ESTATektonParametersestaTektonPipeline.)

The estaTektonPipeline.json is the single file used to configure the build, release and deployment cycles of a software project. It's interpreted by the ESTA Tekton Controller and translated into concrete actions and Tekton pipeline Parameters to start the Tekton Pipelines.

A minimal example is shown in the [Getting Started](file:///C:\display\CLEW\Getting+Started) guide and the schema of this file shall be explained in detail here.

The JSON schema for this file is published at <https://clew-resources.sbb-cloud.net/tekton-schema.json>.

The most stable way in Intellij Idea to support context sensitive completion for YAML and JSON files we suggest to configure the schema globally. Only then it also worked seamlessly for YAML files. As showed in the screenshot. Other IDE's like MS Visual Studio Code support the same feature. You will find setup guide here <https://medium.com/@alexmolev/boost-your-yaml-with-autocompletion-and-validation-b74735268ad7>.



# Pipelines

With the "pipelines" config blocks you can define individual "pipeline profiles" for different cases. Different than the "steps" in estaCloudPipeline.json, pipelines have no fixed type that implies a certain build workflow. The ESTA Tekton pipelines can be configured freely with individual names and settings.

A pipeline block has certain properties to define the trigger criteria and additional properties define the type of the pipeline and it desired output:

**Build pipeline config block** Quelle erweitern

{

"name": "continuous-integration",

"enabled": true,

"triggerType": [

"USER",

"GITEVENT"

],

"branchNamePrefixes": [

"deploy"

],

"versionTagEventPatterns": [

"^(\\d+\\.)(\\d+\\.)(\\\*|\\d+)-SNAPSHOT$"

],

"build": {

"buildDockerImage": true,

"packageAndDeployHelmChart": false,

"additionalBuildParams": "-DskipITs",

   "additionalDeployParams": "-DskipITs",   }

}

### Trigger criteria

* triggerType: defines which events trigger this build step:
  + USER: Control panel action
  + GITEVENT: Bitbucket webhook
  + PIPELINE: Git commit triggered by a Tekton pipeline
* branchNamePrefixes: filter events on Git branches starting with one of the given values
* versionTagEventPatterns: for Git tag events, the tag name needs to match one of the given regex patterns

### Build parameters

The property "build" indicates a build pipeline to run with the following settings:

* buildDockerImage: build a Docker image and push it to Artifactory (unless disabled by deployDockerImage:false)
* packageAndDeployHelmChart: create a Helm chart and push it to the Helm repository in Artifactory
* deployArtifacts: deploy Maven or NPM artifacts to Artifactory

Please note that for each event (Bitbucket webhook, Control panel action or API call) ***all matching*** pipeline profiles will start an according pipeline.

# Stages

The stages configuration represent deployment targets for your applications. The task of actually rolling out a deployment is not part of the ESTA Tekton pipeline but uses [ArgoCD and Helm Charts](file:///C:\display\CLEW\Deployment+with+ArgoCD+and+Helm+Charts) as the only option right now. You can still configure certain aspects of the deployment process in estaTektonPipeline.json which are executed after a successful build.

The "stages" config blocks each describe a deployment target. Besides a unique "stageName", a stage config can contain an "argoCD" and a "helm" block. Here's an example:

**Stage config block** Quelle erweitern

{

"stageName": "dev",

"argoCD": {

"argoCdAppName": "springboot-tekton-dev",

   "autoSync": true

},

"helm": {

"sourceRepository": "https://code.sbb.ch/scm/kd\_example/argocd-apps-helm.git",

"chartFilePath": "clusters/aws01t/values/springboot-tekton.yaml",

"versionProperties": [

".deploymentVersion"

]

}

The "helm" block is merged with the top-level "helm" block described below. If present in a staging block, the staging pipeline will update the release version from the triggering build in the Helm chart (repositoy, chart file) referenced by this config. By default the Helm chart is searched in the project repository unless explicitly specified by the "sourceRepsitory" property.

With the "argoCD" block the staging pipeline is advised to trigger a sync in ArgoCD on the application with the specified name. The hostname of the ArgoCD instance and the access token used for the connection are taken from a Secret stored in the Tekton build namespace.

See the JSON schema or [Parameter Documentation](#ESTATektonParametersestaTektonPipeline.) for detailed description of these properties.

In order to trigger the deployment (staging) after a successful build, define a list of stage names in the pipeline config block using the "stages" property.

# Helm Charts

Tekton build pipelines can also package and deploy [Helm](https://helm.sh/) charts. The "helm" block holds the configuration for this:

**Helm config block** Quelle erweitern

{

"chartFilePath": "./charts/springboot-tekton-app/Chart.yaml",

"chartRepository": "esta.helm.local"

}

See the JSON schema or [Parameter Documentation](#ESTATektonParametersestaTektonPipeline.) for detailed description of these properties.

# Sonar Analysis

Tekton can also be used to perform sonar analyses. The "sonarScan" holds the configuration for this. The sonarScan block can be configured for each build profile separately.

**SonarScan config block** Quelle erweitern

{

"enabled": true,

"projectKey": "springboot-tekton-app"

}

For Maven, NPM Go or Docker projects you can create a sonar-project.properties file to set additional sonar scan analysis parameters according to the official documentation: <https://docs.sonarqube.org/latest/analysis/analysis-parameters/>

This also works for Maven, since we patch the maven configuration according to the sonar-project.properties. Even if you have a sonar-project.properties file, values like sonar.login, sonar.host.url, sonar.organization and so on will be automatically set through the esta-tekton build pipeline.

For Dockerfile analysis, make sure your Dockerfiles are named Dockerfile, Dockerfile.\* or \*.dockerfile to get detected and processed by Sonarqube.

See the JSON schema or [Parameter Documentation](#ESTATektonParametersestaTektonPipeline.) for detailed description of these properties.

# OWASP Dependency Check

You can also perform the OWASP Dependency Check to find vulnerabilities in your software. It is configured in the "owaspDependencyCheck" block. You can configure it for Javascript and Java projects.

The check is documented here: <https://jeremylong.github.io/DependencyCheck/index.html>

**OWASP Dependency Check config block** Quelle erweitern

{

"enabled": true

"additionalParams": "--failOnCVSS 5"

}

# Docker Settings

Building and publishing Docker images as artifacts of the build process requires some configuration defined in the "docker" block. We support one single Docker build blocks but also multiple blocks which are defined as an array. Following some examples:

**Docker config block** Quelle erweitern

{

"artifactoryDockerRepo": "esta.docker",

"imageName": "springboot-tekton-app",

"dockerFile": "docker/Dockerfile"

}

Alternatively you can also use multiple Docker Build Blocks.

**Docker config block** Quelle erweitern

[

{

"artifactoryDockerRepo": "esta.docker",

"imageName": "springboot-tekton-app",

"dockerFile": "docker/Dockerfile.spring"

},

{

"artifactoryDockerRepo": "esta.docker",

"imageName": "another-springboot-tekton-app",

"dockerFile": "docker/Dockerfile.other"

}

]

These docker builds run in parralel. If one docker image builds on top of another, specify that dependency with the runAfterImageName property referencing a imageName value.

See the JSON schema or [Parameter Documentation](#ESTATektonParametersestaTektonPipeline.) for detailed description of these properties.

# Examples

A collection of example estaTektonPipeline.json files is maintained in [this repository](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse).

### Java (Maven)

For Java projects the ESTA Tekton system currently only supports Maven. In JSON add a "mvn" block to define a Maven build.  
A project can either be configured to push a Docker image and/or a JAR file to Artifactory as build artifacts.

* [SpringBoot application with Dockerfile and SonarScan](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/java/springboot/estaTektonPipeline.json)
* [Library/JAR project](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/java/library/estaTektonPipeline.json)
* [Nightly builds with Continuous Delivery](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/java/continuous/estaTektonPipeline.json)

Note the difference!

When using a custom Dockerfile you may need to replace certain commands to work with Tekton. [Read more about Dockerfile Migration](file:///C:\display\CLEW\Migrating+from+ESTA+Cloud+Pipeline).

### Node.js / NPM

Node.js projects can be built with npm commands. To indicate an NPM project, add a "npm" block in JSON.  
The typical use case is for Angular applications which are compiled to static files and packed into an ngnix Docker image for serving.

* [Angular application (Nginx Docker image)](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/node/angular/estaTektonPipeline.json)
* [Angular application published on S3](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/node/angular-on-s3/)
* [Node.js module](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/node/module/estaTektonPipeline.json)

### Golang

* [Go application](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/go/application/estaTektonPipeline.json)

### Docker

Pure Docker image builds can be done with Tekton, too. All it needs is a "docker" block in JSON pointing to the Dockerfile(s) to use and to which Artifactory docker repository the images shall be pushed.

* [Simple Docker image build](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/docker/simple)
* [Multiple Docker images](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-json-examples/browse/docker/multi)

### Parametrized Builds

Since version 0.27.x you can start builds with parameters. In the UI there will be just a text field, which shows the parameters as a JSON value prefilled. In future versions you will have better UI support for this feature. The JSON will be not validated against the parameter definition.

You find an example here. Also how you can iterate over the variables.

<https://code.sbb.ch/projects/KD_ESTA_TEST/repos/esta-tekton-helm-sample/browse/estaTektonPipeline.yaml#41>

Or this is how you define the parameters globally in your estaTektonPipeline.json.

"parameters": [

{

"name": "env",

"type": "Choice",

"values": ["dev"],

"label": "Environment",

"description": "Target environment",

"options": ["dev", "test", "prod"]

},

{

"name": "notificationEmails",

"type": "Multichoice",

"values": ["john.doe@sbb.ch", "jane.doe@sbb.ch"],

"options": ["john.doe@sbb.ch", "jane.doe@sbb.ch", "some-one.else@sbb.ch"]

"description": "The email addresses which the notifications are sent to"

},

  {

"name": "notificationSubject",

"type": "String",

"values": ["New release deployed"],

"description": "Subject of the notification email"

}

],

Details about the configuration you can find in the parameter documentation.

Upon pipeline execution, the parameter values will be exposed to each pipeline task as JSON formatted string via the environment variable USER\_PIPELINE\_PARAMETERS\_JSON.

Also each parameter value get's exposed as a environment variable in the pod by it's name:

The parameter name "env" get's exposed as environment variable UP\_ENV. With UP\_ENV\_SIZE you get the size of the array values. With UP\_ENV\_0, UP\_ENV\_1 you access it's values.  
"notificationEmails" will be written in the environment variable UP\_NOTIFICATIONEMAILS.

You find an example here. Also how you can iterate over the variables.

<https://code.sbb.ch/projects/KD_ESTA_TEST/repos/esta-tekton-helm-sample/browse/estaTektonPipeline.yaml#41>

### Custom Pipelines

This is an experimental feature of ESTA Tekton. Read more in the [Advanced Topics](file:///C:\display\CLEW\Advanced+Topics#AdvancedTopics-Custompipelines).

* [Build multiple docker images](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-builder-image/browse/estaTektonPipeline.json)

# Parameter Documentation

This is the complete list of JSON properties supported by the estaTektonPipeline.json schema in the latest stable version.

### Global Properties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **productName** | Name of the product/application. Defaults to the git repository name if not set. | no | Git repository name |  |
| **docker** | Defines the output of the Docker images. See [Docker configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **helm** | Configures the creation and deployment of Helm charts. See [Helm configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **builder** | Configures the builders like java, kotlin, python, node, go  See [Builder configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **mvn** | Settings for Java builds using Maven. See [Maven configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **npm** | Settings for Node.js NPM builds. See [NPM configuration structure](https://confluence.sbb.ch/display/CLEW/ESTA+Tekton+Parameters+-+estaTektonPipeline.json#ESTATektonParametersestaTektonPipeline.json-NPMconfigurationstructure) for details. | no |  |  |
| **flutter** | Settings for Flutter builds. See [Flutter configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **go** | Settings for Golang builds. See [Go configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **buildpack** | Settings for container image builds with [Buildpacks](https://buildpacks.io). See [Buildpack configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **pipelines** | Defines a list of pipeline configurations. See [Pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no | [] |  |
| **stages** | Defines a list of staging/deployment configurations. See [Stage configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no | [] |  |
| **notifications** | Defines a list of notification settings. See [Notification configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no | [] |  |
| **advancedBuildSettings** | Defines the advanced build settings.  See [Advanced Build Settings](#ESTATektonParametersestaTektonPipeline.) for details. | no | [] |  |

### Docker configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **artifactoryDockerRepo** | Repository where the Docker image should be pushed to Must include the suffix ".docker" | yes |  |  |
| **dockerFile** | The path to the Dockerfile e.g. ./Dockerfile or docker/Dockerfile If the given Dockerfile is not found for Maven projects, a Selfrunning Jar build is started where a Dockerfile is automatically generated. | no | "./Dockerfile" |  |
| **imageName** | Name of the docker image to create. If omitted, the productName property will be used. | no | .productName |  |
| **baseImageFrom** | The base docker image used in the Selfrunning Jar build case | no | "esta.docker.bin.sbb.ch/esta/ubi9/java-17:latest" |  |
| **contextDir** | Directory for the Docker build to be executed from.  The path to the Dockerfile as well as all files referenced from within the Dockerfile are resolved from this directory. | no | "." |  |
| **ocBuildVars** | List of own image build arguments in the KEY=value format. Extra values passed as build args to the build command. Use them with ARG KEY inside the Dockerfile. | no | [] |  |
| **pomArtifactId** | In case of a Maven multi-module project, you can choose the artifactId which is used to build the image. The property has no effect in case of a Selfrunning Jar build (see dockerFile property). | no |  |  |

If your Dockerfile is not on the root folder and you're using copy steps in it, contextDir must be set and define the directory on which the Dockerfile is located! Otherwise the build will fail as it can't find the content to copy into the image during the build.

### Helm configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **chartFilePath** | This is the path from your root folder to your Chart file. For example: ./charts/your-repo/Chart.yaml | no |  |  |
| **chartRepository** | The name of the Helm repository in Artifactory. If specified the chart is packaged and uploaded to this repository | no |  |  |
| **versionProperties** | List of properties of the Helm chart file to set the release version to. Each property is defined as a yq expression (see [documentation](https://github.com/mikefarah/yq#yq)) which is similar to [jq syntax](https://jqlang.github.io/jq/manual/).  When used in .stages[\*].helm config block you can use the following code segment to update the dependecies in an Umbrella-Chart:  "versionProperties": [  ".dependencies[] | select(.name==\"name-of-your-project\").version"  ] | no | ".version" |  |
| **sourceRepository** | The Git repository where Helm charts for staging/deployment are maintained. Only allowed in .stages[\*].helm config block. Default is the current Git repository. | no |  |  |
| **sourceBranchOrTag** | Git branch where the Helm charts should be updated. Only allowed in .stages[\*].helm config block. | no | "master" |  |
| **lintArgs** | Additional arguments for the helm lint command | no |  |  |

### Builder configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **java** | Java major version you want to use to build: 11, 17, 21, 17-Graal, 21-Graal | no | "17" |  |
| **maven** | Maven minor version you want to use to build: 3.8, 3.9 | no | "3.8" |  |
| **kotlin** | Kotlin minor version you want to use to build: 1.7 | no | "1.7" |  |
| **node** | Node major version you want to use to build: 14, 16, 18, 20 | no | "18" |  |
| **python** | Python minor version you want to use to build: 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 | no | "3.10" |  |
| **go** | Go minor version you want to use to build: 1.16, 1.18, 1.19, 1.20, 1.21, 1.22 | no | "1.18" |  |
| **customBuilderImage** | You can define your own custom builder image which behaves the same as the esta-builder-image but has for example different sdk versions preinstalled | no |  |  |

### Maven configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **parentPom** | Path to the parent pom file. In most cases you don't need to set this. By default the pom.xml is taken. But if you have a parent pom.xml in another subfolder, you can configure it here. | no | "pom.xml" |  |
| **deployableArtifactsPom** | Path to pom file which specifies the deployable jar artifacts for Artifactory. This has no direct influence on the Docker build.  This property is not meant to determine the artifact which get's packed into the Docker image. | no | "pom.xml" |  |
| **artifactoryMavenRepo** | The Maven repository in Artifactory where the Maven artifacts will be published to. Required when *deployArtifacts* in step config is enabled. Must end with the suffix ".mvn" | no |  |  |
| **isTychoBuild** | Check if Repository use Tycho Build | no | false |  |
| **forceUnstableBuild** | Let builds succeed even if tests fail but report pipeline status as UNSTABLE | no | false |  |
| **buildTaskName** | Name/variant of the Tekton task used to perform the build step. Default tasks available for selection:   * "esta-maven-build": 2G memory; max. 4G memory and 2 CPU * "esta-maven-build-small": 600M memory; max. 1.5G memory and 1 CPU * "esta-maven-build-large": 6G memory; max. 9G memory 4 CPU * "esta-maven-build-extra-large": 20G memory; max. 24G memory and 6 CPU | no | "esta-maven-build" | 0.24.0 |

### NPM configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **packageJson** | Specifies the path where the package.json is located. Needs to end with 'package.json' | no | "./package.json" |  |
| **publishablePackageJsons** | Specifies the package.json files for publishable projects | no |  |  |
| **additionalNpmScripts** | Additional NPM commands to run at build time as specified in package.json. Multiple commands can be provided separated by a space. | no |  |  |
| **targetRepo** | The target repository in Artifactory where the NPM artifacts will be published to. Required when *deployArtifacts* in step config is enabled. | no |  |  |
| **buildTaskName** | Name/variant of the Tekton task used to perform the build step. Default tasks available for selection:   * "esta-npm-build": 3G memory; max. 6G memory and 2 CPU * "esta-npm-build-large": 5G memory; max. 9G memory 4 CPU * "esta-npm-build-extra-large": 12G memory; max. 20G memory and 6 CPU | no | "esta-npm-build" | 0.25.2 |

### Flutter configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **buildScripts** | List of build scripts which should run in the flutter build task. For example:  ["prepare-zip.sh", "prepare-dev-zip.sh"] | no | prepare-zip.sh |  |
| **pubspecYamlPath** | The path to the flutter pubspec.yaml file | no | pubspec.yaml |  |

### Go configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **buildPackages** | The packages to build | no | "./cmd/..." |  |
| **useMage** | Enforce use of the build tool mage | no | false |  |
| **additionalGoParams** | Go build orders. Multiple commands can be separated by spaces. | no |  |  |

### Python configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **targetRepo** | The target repository in Artifactory where the Python artifacts will be published to. Required when *deployArtifacts* in step config is enabled. | no |  |  |
| **additionalPythonParams** | Multiple commands can be separated by spaces. | no |  |  |

### Parameters configuration structure

The full functional parameter UI will be released in a future release as for the moment, we display a simple json structure in the UI.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **name** | A unique name for this parameter. i.E. environments, emailadresses, and so on. | yes |  | 0.27.x |
| **type** | Type of the parameter to define how the parameter will be displayed in the UI and processed in the templates. Possible values are: String, Multiline, Choice, Multichoice, Boolean | yes |  | 0.27.x |
| **options** | The options which will be displayed in the UI to select from. Here you can define for example for a parameter named environment the options: dev, test , int, prod which will be then displayed as a multichoice or choice checkbox or radiobutton selection control in the UI. | no |  | 0.27.x |
| **values** | The values array holds the preselected or predefined values which will be displayed in the UI. If you have the options dev, test , int, prod you can for example specify the preselected values: dev, test | no |  | 0.27.x |
| **label** | The labels value defines the label which is displayed in the UI. If not defined, the name will be used as label in the UI. | no |  | 0.27.x |
| **description** | Just to describe the parameter. This will be shown as a tooltip in the ui when hovering over the question mark in the UI. | no |  | 0.27.x |

### Buildpack configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **builder** | The builder image to use. Currently [Paketo](https://paketo.io/) and [Heroku](https://devcenter.heroku.com/articles/buildpacks) buildpack builders are supported | no | "paketobuildpacks/builder:base" |  |
| **envVars** | A list of environment variables to be set for the Buildpack lifecycle process. Specify them as strings of the format "VAR\_NAME=value". | no | [] |  |

### Pipeline configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **name** | Unique name for this pipeline config. You can choose your own names. | yes |  |  |
| **enabled** | Should this build profile be switched on or off (true / false) | no | true |  |
| **triggerType** | List of triggers that will start this pipeline. "USER": is triggered through the REST API, usually from the Tekton Control Panel "GITEVENT": is triggered by pushes (branch or tag) to the Git repository "PIPELINE": is triggered by Git branch pushes issued by a Tekton pipeline "CRON": is triggered by the ESTA Tekton Controller at the time configuered by cronTab. The CRON configuration will be only read from the default branch (main, master, ...) | no | "USER" |  |
| **branchNamePrefixes** | List of prefixes of Git branches on which the pipeline should be executed | no |  |  |
| **branchNamePatterns** | List of regex patterns to match or not match Git branches on which the pipeline should be executed. Has precedence over branchNamePatterns. | no |  |  |
| **versionTagEventPatterns** | List of regex patterns for Git tags when pushed to Bitbucket. A regex like "^(\\d+\\.)?(\\d+\\.)?(\\\*|\\d+)$" would match a tag like 10.5.100 | no |  |  |
| **cron** | Unix-style [Cron expression](https://en.wikipedia.org/wiki/Cron) to define times when automated pipeline runs should start. The cron configuration will be only read from the default branch (master, main, ...) Required in conjunction with triggerType "CRON". Please note: nicknames like "@midnight" are not supported. | no |  |  |
| **stages** | Define a list of stage names to trigger if pipeline run was successful | no | [] |  |
| **build** | Start a ***build*** pipeline with these settings. See [Build pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **tagging** | Start a ***tagging*** pipeline with these settings. See [Tagging pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **custom** | Start a custom pipeline with these settings. See [Custom pipeline configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **envSecret** | OpenShift secret name holding env variables to be exposed in pipeline steps. The name needs to start with "pipeline-env". | no | "esta-tekton-pipeline-env" |  |
| **tasks** | List of additional tasks to be inserted into the pipeline. See [Custom task configuration structure](#ESTATektonParametersestaTektonPipeline.) for details and [README.md](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-templates/browse/tasks/README.md) for task documentation. | no | [] |  |
| **failOnQualityGateFailure** | Whether to fail the pipeline if the quality gate fails. | no | false |  |

### Build pipeline configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **buildDockerImage** | Build a Docker image | no | false (true for docker-only repositories) | |
| **deployDockerImage** | Push the built Docker image to Artifactory | no | true |  |
| **deployArtifacts** | Should the generated artifacts (Mvn, NPM, Go) be deployed to Artifactory | no | false |  |
| **packageAndDeployHelmChart** | Package a Helm chart and deploy it to Artifactory | no | false |  |
| **additionalBuildParams** | Additional parameters appended to commands that execute builds (mvn, npm run, go build) | no |  |  |
| **additionalDeployParams** | Additional deploy parameters added to Maven commands. | no |  |  |
| **additionalDockerImageTags** | List of additional tags the Docker image shall be pushed to | no | [] |  |
| **sonarScan** | You can configure the sonarScan for each build step separately. A simple example looks like this:  "sonarScan": {    "enabled": true }  For further information see the sonar configuration structure description below or the sonar details above: [SonarAnalysis](#ESTATektonParametersestaTektonPipeline.) | no |  |  |
| **owaspDependencyCheck** | You can configure the owaspDependencyCheck for each build step separately. A simple example looks like this:  "owaspDependencyCheck": {    "enabled": true }  For further information see the owasp dependecy check configuration structure description below or the details above: [OWASPDependencyCheck](#ESTATektonParametersestaTektonPipeline.) | no |  |  |
| **gitguardian** | Configure GitGuardian secrets scan during build. For details and further configuration options see the [GitGuardian secrets scan configuration structure](#ESTATektonParametersestaTektonPipeline.) | no |  | 0.28.0 |
| **testcontainers** | Enable and configure an environment for running [Testcontainers](https://testcontainers.com/) in the build step. Enabling testcontainers looks like this:  "testcontainers": {    "enabled": true }  For details and further configuration options see the [Testcontainers configuration structure](#ESTATektonParametersestaTektonPipeline.) | no |  |  |
| **testreport** | Enable and configure visual test reports creation. For details and confguration options see the [Testreport configuration structure](#ESTATektonParametersestaTektonPipeline.) and [this how-to article](file:///C:\display\CLEW\Enabling+Test+Reports+in+ESTA+Tekton). | no |  |  |
| **versionTag** | Version tag to use for docker builds with this profile. If set, this overrides the version tag derrived from the Git tag or project meta files. Can contain placeholders for [pre-defined pipeline parameters](#ESTATektonParametersestaTektonPipeline.) like "snapshot-${DATE}" or "build-${BRANCH\_LABEL}" | no |  |  |
| **docker** | Override the top-level docker build configuration. See [Docker configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **helm** | Override the top-level Helm charts configuration. See [Helm configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **mvn** | Override the top-level Maven build settings. See [Maven configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **npm** | Override the top-level settings for NPM builds. See [NPM configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **flutter** | Override the top-level settings for Flutter builds. See [Flutter configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |
| **go** | Override the top-level settings for Golang builds. See [Go configuration structure](#ESTATektonParametersestaTektonPipeline.) for details. | no |  |  |

### Tagging pipeline configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **versionIncrementPattern** | Regex pattern defining how the version is incremented. The first segment in brackets will be incremented by one and the following brackets will be set to 0. Example: "REL-\\d+\\.(\\d+)\\.(\\d+)" – this will increment "REL-2.1.13" to "REL-2.2.0" | no | "^(\\d+)\\.(\\d+)\\.(\\d+)" |  |
| **versionIncrementPosition** | Which segment (major, minor, patch) to increment from the versionIncrementPattern.  If not using strict semantic versioning, the position can be denoted with "one", "two", "three", "four" or "five". The used number must match the number of the referenced pair of braces on versionIncrementPattern!  To disable this functionality you can use `none`. | no | "minor" |  |
| **preReleaseVersionIncrement** | Should the next pre-release version (-SNAPSHOT) be set on automatic increment and which position (major, minor, patch). This option is only considered for Maven projects using strict [sematic versioning](https://semver.org/). | no |  |  |
| **updateHelmChart** | Should Helm charts (if defined in helm block) be updated in this tagging pipeline | no | true |  |

For pure Maven builds, these settings should be something like this:

"tagging": {

"updateHelmChart": true,

"versionIncrementPosition": "none",

"preReleaseVersionIncrement": "minor"

},

With these settings the release build will just remove the -SNAPSHOT suffix, increment the version after the releasebuild and add -SNAPSHOT suffix again.

### Custom pipeline configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **pipelineTemplate** | Name of an OpenShift template defining a custom PipelineRun to process for this step. When ending with .yaml, the template is resolved to a file located in the 'tekton' folder of this repository. | no |  |  |
| **pipelineRef** | Name of a custom Pipeline to run for this step. A Tekton pipeline with the given name must be defined in the build namespace. | no |  |  |
| **params** | Map (key: value) of parameters passed to the PipelineRun template. Values can contain placeholders for [pre-defined pipeline parameters](#ESTATektonParametersestaTektonPipeline.) like "rel-${VERSION\_TAG}". | no | {} |  |

### Sonar configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **enabled** | This flag decides whether a sonar analysis is performed or not. Please also consult the details: [SonarAnalysis](#ESTATektonParametersestaTektonPipeline.)  If you want to do Sonarscan in a release build, don't forget to set the **releaseBranchName**! | yes | false |  |
| **projectKey** | The projectKey is the project into which the analysis report is loaded. | no | .productName |  |
| **~~additionalParams~~** | Additional maven options like "-DskipTests=true" **DEPRECATED!** Set Sonar options for Maven builds to build.additionalProperties | no |  | 0.25.0 (deprecated) |
| **releaseBranchName** | Specify the branch name used for SonarQube in case there is no git branch in the pipeline context. For all builds based on a tag this property is required to execute the sonar scan. | no |  |  |

### GitGuardian secrets scan configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **enabled** | Enable the [GitGuardian](file:///C:\display\CYBERSEC\GitGuardian) secrets scan on the current repository | yes | false | 0.28.0 |
| **reportMode** | How to report GitGuardian warnings. FAILED: make the pipeline fail with an error UNSTABLE: report the pipeline to be unstable LOG: only print the findings in the log | no | "UNSTABLE" | 0.28.0 |
| **secretName** | Alternative secret name to read GitGuardian credentials from. Must contain entries named GITGUARDIAN\_API\_KEY and GITGUARDIAN\_INSTANCE | no |  | 0.28.0 |

### OWASP dependency check configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **enabled** | Do you want run the OWASP Dependency Check? (true / false) The OWASP Dependency Check will by default just generate a report which can be displayed in Sonarqube. You need to click in Sonarqube on "More" → "Dependency Check". The OWASP Dependency Check Report will be displayed. If you want to fail the build if the dependency check finds too many issues, you need to set the according property in additionalParams. See the following documentation of the parameter. | yes | false |  |
| **additionalParams** | Additional Parameter options like "--failOnCVSS 5 --disableRetireJS". You can find the possible parameter documentation on: <https://jeremylong.github.io/DependencyCheck/dependency-check-cli/arguments.html> | no |  |  |

### Testcontainers configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **enabled** | Enable pipeline environment for running Testcontainers. This option only has an effect in Maven, Npm and Go projects. | yes | false |  |
| **kubedockParams** | Additional arguments passed to the kubedock server command in order to tweak the containers environment. See the [Kubedock documentation](https://github.com/joyrex2001/kubedock#containers) for details. | no |  |  |

### Testreport configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **enabled** | Enable the creation of test reports. This option only has an effect in Maven, Npm and Go projects. | yes | false |  |
| **artifactoryRepository** | Name of the Artifactory repository where the generated reports are uploaded to. Must end with .generic | no | "esta-tekton.generic" |  |
| **resultsDir** | Location where the result XMLs are located, from which the report should be created. | no | "target/surefire-reports" |  |
| **reportQualifiers** | Additional qualifiers (aliases) to store the generated report with. These qualifiers can be used to connect new reports with in order to show Trend and history. | no | [] |  |
| **previousQualifiers** | Copy history from the last report generated with these qualifiers to make the trend and history graphs working. The first matching qualifier will be used. | no | [] |  |

### Custom task configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **name** | Name of this task. Needs to be unique within the pipeline spec. | yes |  |  |
| **taskRef** | Name of the Tekton task to execute. The according task needs to be present in the build namespace. See [README.md](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-templates/browse/tasks/README.md) for documentation of available tasks. | yes |  |  |
| **params** | Map (key: value) of parameters passed to the pipeline task. Values can contain placeholders for [pre-defined pipeline parameters](#ESTATektonParametersestaTektonPipeline.) like "rel-${VERSION\_TAG}". | no | {} |  |
| **executionMode** | Mode how this task is executed. REGULAR = injected into the regular pipeline execution (further specify with runAfter and runBefore) FINALLY = runs in the finally block (guarded with runConditions) | no | "REGULAR" |  |
| **runBefore** | Name(s) of the pipeline task(s) this task must be preceded. Look at past pipeline runs to see the names and order of the standard pipeline tasks. Only supported with executionMode "REGULAR". | no |  |  |
| **runAfter** | Name(s) of the pipeline task(s) this task shall execute after. If omitted, the task is appended to the end of the pipeline. Only supported with executionMode "REGULAR". | no |  |  |
| **runConditions** | Condition(s) when this task should run. Use short condition aliases "success()", "failure()", "always()" or expressions like "$(tasks.build.status) in None,Succeed". You can use [Tekton variables](https://tekton.dev/docs/pipelines/variables/) in these expressions. Only supported with executionMode "FINALLY". | no | "success()" |  |

### Stage configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **stageName** | Unique name for this stage config | yes |  |  |
| **argoCD** | ArgoCD configuration structure used for deyploment. See [ArgoCD configuration structure](#ESTATektonParametersestaTektonPipeline.) below. | no |  |  |
| **helm** | Helm configuration defining charts used for deployment. These stage settings overwrite properties from the global Helm configuration structure. See [Helm configuration structure](#ESTATektonParametersestaTektonPipeline.) below. | no |  |  |

### ArgoCD configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **argoCdAppName** | The app name which will be referenced in ArgoCD. Default value is the top-level productName property | yes | .productName |  |
| **autoSync** | If true, the deployment pipeline will initiate an ArgoCD sync | no | false |  |
| **argoCdSecret** | OpenShift secret name holding env variables for the argocd task | no | "argocd-env-secret" |  |
| **argoCdAppSyncFlags** | ArgoCd app sync flags i.e. --force --replace, with which you can control the way your app should be synced by ArgoCD. You find the flags documented here: <https://argo-cd.readthedocs.io/en/stable/user-guide/commands/argocd_app_sync/>  For example if you are using --force --replace, it would redeploy also in SNAPSHOT case. | no |  |  |

### Notification configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **type** | Type/delivery of notification messages. Currently supported are "EMAIL" or "TEAMS" | yes |  |  |
| **enabled** | Enable or disable sending of notifications | no | true |  |
| **eventTypes** | Filter for events that trigger this notification. Possible values are "SUCCESS", "FAILURE", "ABORTED". If empty, all events will trigger notifications. | no |  |  |
| **pipelineTypes** | Filter for pipeline types to trigger this notification. Possible values are "TAGGING", "BUILD", "STAGING", "CUSTOM"). If empty, all pipeline types will trigger notifications. | no |  |  |
| **branchNameExcludes** | List of prefixes of Git branches on which NO notifications shall be sent | no | [] |  |
| **recipients** | List of recipient email addresses for EMAIL type notifications. Use "$actor" as a placeholder for the user who triggered the pipeline event. | (yes) | [] |  |
| **~~webhookUrl~~** | Webhook URL for TEAMS type notifications. DEPRECATED. Save the webhook URL to a secret and reference with webhookSecretName and webhookUrlSecretKey instead. | (yes) |  | Deprecated in 0.24.0 |
| **webhookSecretName** | Secret name providing the webhook URL for TEAMS type notifications | (yes) |  | 0.24.0 |
| **webhookUrlSecretKey** | Key of the secret entry holding the webhook URL for TEAMS type notifications | no | "URL" | 0.24.0 |
| **title** | Override default title/subject for this notification. See the [supported placeholders](file:///C:\display\CLEW\ESTA+Tekton+Workflow#ESTATektonWorkflow-Variablesubstitution) that will be replaced with the actual values. | no |  |  |
| **message** | Override default message text for this notification. See the [supported placeholders](file:///C:\display\CLEW\ESTA+Tekton+Workflow#ESTATektonWorkflow-Variablesubstitution) that will be replaced with the actual values. | no |  |  |

### Advanced Build Settings configuration structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property name** | **Description** | **Required** | **Default value** | **Since version** |
| **buildWorkspaceSize** | Here you can define the **build** workspace size. You need to do that in case your build process produces big artifacts and the default size of the build pipeline workspace is not big enough. You can set something like 2Gi or 500Mi as a value. | no | 4Gi (mvn) 3Gi (npm) 3Gi (python) 2Gi (go) 1Gi (others) |  |
| **taggingWorkspaceSize** | Here you can define the **tagging** workspace size. You need to do that in case your git repository contains large files and the default size of the tagging pipeline workspace is not big enough. You can set something like 2Gi or 500Mi as a value. | no | 500Mi |  |
| **buildCacheSize** | Sets the size of the cache volume used for build pipelines. The cache serves as a persistent store for downloaded dependencies. Depending on the application the size of all dependencies can vary and therfore the cache size can be adjusted individually in case you see disk space errors. Set values like 800Mi or 6Gi. | no | 2Gi |  |
| **pipelineTimeout** | This sets the global timeout of the Pipeline. The Default of Tekton Pipelines is 1h0m0s which stands for 1 hours 0 minutes 0 seconds. So you could set for example a global timeout to 2h30m0s. | no | no value set (1h0m0s) |  |
| **taskTimeouts** | This sets the timeout of specific tasks. The default for each is task is 1h0m0s which stands for 1 hours 0 minutes 0 seconds. The values are to be set in an object. The name of the task is the key, and the timeout time is the value:  "taskTimeouts": {    "docker-build": "2h0m0s" } | no | no value set  (every task 1h0m0s) | 0.28.0 |

# Pre-defined Pipeline Parameters

The following parameters are provided by the ESTA Tekton Controller and passed to pipeline templates when starting a pipeline. They can be used in custom pipeline templates as well as for placeholder substitution in values from estaTektonPipeline.json. Insert these parameters in JSON with this syntax: ${PARAM\_NAME}

**REPOSITORY** - Bitbucket repository key in the form <project-key>--<repository-slug> (all lower-cased)

**PRODUCT** - The productName from estaTektonPipeline.json or the repository slug if not defined

**RANDOMNAME** - A random alpha-numeric value of 6 characters

**GIT\_URL** - The git clone URL to the source repository

**GIT\_REVISION** - The git revision (sha) to be cloned into the pipeline workspace

**GIT\_REV\_SHORT** - The short version of the GIT\_REVISION (first 10 characters)

**GIT\_COMMIT\_MESSAGE**- The first line of the last commit message

**GIT\_BRANCH** - Git branch name (not set when pipeline started from a Git tag)

**DATE** - The current date of the pipeline execution (yyyyMMdd)

**DATETIME** - The current date/time of the pipeline execution (yyyyMMddHHmmss)

**ACTOR** - Email address of the person/account who initiated the pipeline run

**USER\_ID** - User ID (SBB employee number/unique identifier for system users) of the person/account who initiated the pipeline (since 0.28.0)

**VERSION\_TAG** - Release version (for tagging) or Git tag (for release builds) – ATTENTION: This parameter is not available for builds based on a branch

**BRANCH\_LABEL** - Normalized name of the Git branch (lower-cased, alpha-numeric)

**CI**- Could be used to determine if running on Tekton or not. Comparable to former check for BUILD\_NUMBER on Jenkins based build systems.

**PACKAGE\_VERSION**- Package Version is the current Version. The Version is written in pom.xml, package.json or estaTektonPipeline.json