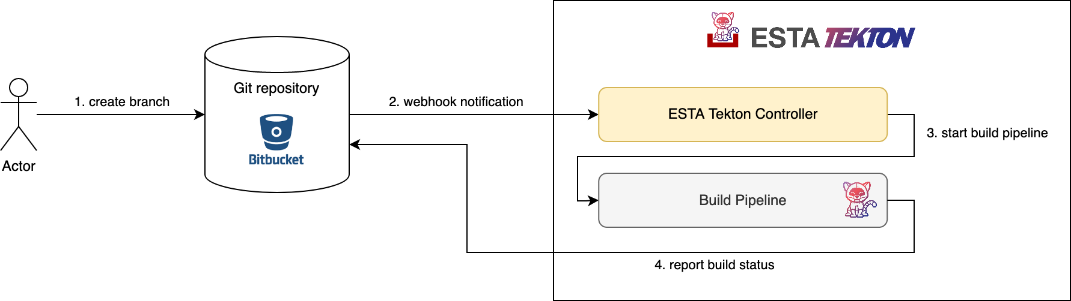
# ESTA Tekton Workflow

* [Continuous Integration](#ESTATektonWorkflow-ContinuousIntegratio)
  + [Scheduled/Nightly Builds](#ESTATektonWorkflow-Scheduled/NightlyBui)
  + [Snapshot Deployments](#ESTATektonWorkflow-SnapshotDeployments)
* [Release Process](#ESTATektonWorkflow-ReleaseProcess)
  + [Auto-releasing from Git Commits](#ESTATektonWorkflow-Auto-releasingfromGi)
* [Bitbucket Integration](#ESTATektonWorkflow-BitbucketIntegration)
  + [Project Scanner](#ESTATektonWorkflow-ProjectScanner)
  + [Build Status Notifications](#ESTATektonWorkflow-BuildStatusNotificat)
* [Notifications](#ESTATektonWorkflow-Notifications)
  + [Variable substitution](#ESTATektonWorkflow-Variablesubstitution)
  + [Email Notifications](#ESTATektonWorkflow-EmailNotifications)
  + [MS Teams Notifications](#ESTATektonWorkflow-MSTeamsNotifications)
  + [Customize Default Notification Texts](#ESTATektonWorkflow-CustomizeDefaultNoti)

# Continuous Integration

CI builds are usually triggered by pushed to the Bitbucket repository and shall not create any artifacts but simply build the application and run unit tests. The build status is then reported to Bitbucket and other recipients.



1. Create a development branch in Git
2. Bitbucket webhook triggers continuous build in ESTA Tekton Controller
3. Build pipeline runs and executes tests
4. Tekton reports build status to Bitbucket and configured recipients

## Scheduled/Nightly Builds

The ESTA Tekton system also allows to define scheduled pipeline runs via the estaTektonPipeline.json. This is done with a pipeline config block that includes "CRON" in the triggerType list and defines the schedule when to run start the pipeline with a [cron expression](https://crontab.guru/). Source for the (build) pipeline run is the first entry from the branchNamePrefixes list or main branch if not specified.

**Scheduled pipeline configuration example** Quelle erweitern

{

...

"pipelines": [

...

{

"name": "nightly",

"cron": "15 4 \* \* 1-5",

"triggerType": [

"CRON"

],

"branchNamePrefixes": [

"master"

],

"build": {

"buildDockerImage": true,

"versionTag": "nightly-${DATE}"

}

}

]

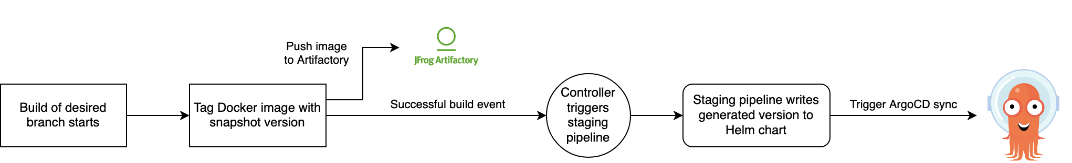
}

The example above starts a build pipeline at 04:15 on every day-of-week from Monday through Friday which builds a Docker image and assigns it the version tag "nightly-YYYYMMDD".

Scheduled pipelines and the time of their next execution are listed in the repository details inside the Tekton Control Panel.

## Snapshot Deployments

Although ESTA Tekton suggests to create [releases](#ESTATektonWorkflow-ReleaseProcess) whenever you want to deploy an application or a service, the scenario of continuously deploying snapshots of the current state of development is real. The workflow could look as follows:



In order to make the deployment of snapshot builds via ArgoCD actually work, it's recommended to assign unique versions to each build. As a source for uniqueness one can use the current Git hash and/or the date/time of the build. Use the "versionTag" property with [pre-defined parameters](file:///C:\display\CLEW\ESTA+Tekton+Parameters+-+estaTektonPipeline.json#ESTATektonParametersestaTektonPipeline.json-Pre-definedPipelineParameters) in your pipeline config to compose such tags under which the build artifacts are published:

**Continuous snapshot pipeline configuration** Quelle erweitern

{

...

"pipelines": [

{

"name": "continuous-snapshot",

"triggerType": [

"GITEVENT",

"USER"

],

"branchNamePrefixes": [

"develop"

],

"build": {

"buildDockerImage": true,

"versionTag": "snapshot-rev-${GIT\_REV\_SHORT}"

"sonarScan": {

"enabled": true

}

},

"stages": [

"dev"

]

}

],

"stages": [

{

"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"

]

}

}

]

This example will build a Docker image with a tag like snapshot-rev-1405327b60a and then update the deploymentVersion value of an ArgoCD source repository for automatic deployment.

Attention: Helm chart versioning

When using "packageAndDeployHelmChart":true in your pipeline config, the Helm chart from your repository will also be released with the version specified by the "versionTag" property. Please note that Helm requires versions that follow the [semver pattern](https://helm.sh/docs/chart_best_practices/conventions/#version-numbers). Therefore the composition of the version needs to be adapted to produce valid semantic versions. Here are some examples that work:

* "0.0.0-snapshot-rev-${GIT\_REV\_SHORT}"
* "0.0.${DATE}-snapshot-${GIT\_REV\_SHORT}"
* "${DATETIME}-snapshot.rev-${GIT\_REV\_SHORT}"

### Using the package version

If the "versionTag" property is omitted, the build pipeline will take the version from package files like pom.xml or package.json and use this for the build artifacts. Because the version doesn't change on each build, deployment via ArgoCD will not work out of the box because ArgoCD will not detect any changes in the Helm chart. This can still work but requires a little hack inside the Helm chart used for deploying the application: add a random value to the Deployment resource template. With this, ArgoCD will detect changes on each refresh operation and sync the changed resource to Openshift.

**Deployment resource template with random value** Quelle erweitern

...

spec:

template:

metadata:

{{- if hasSuffix "-SNAPSHOT" .Values.deploymentVersion }}

    annotations:

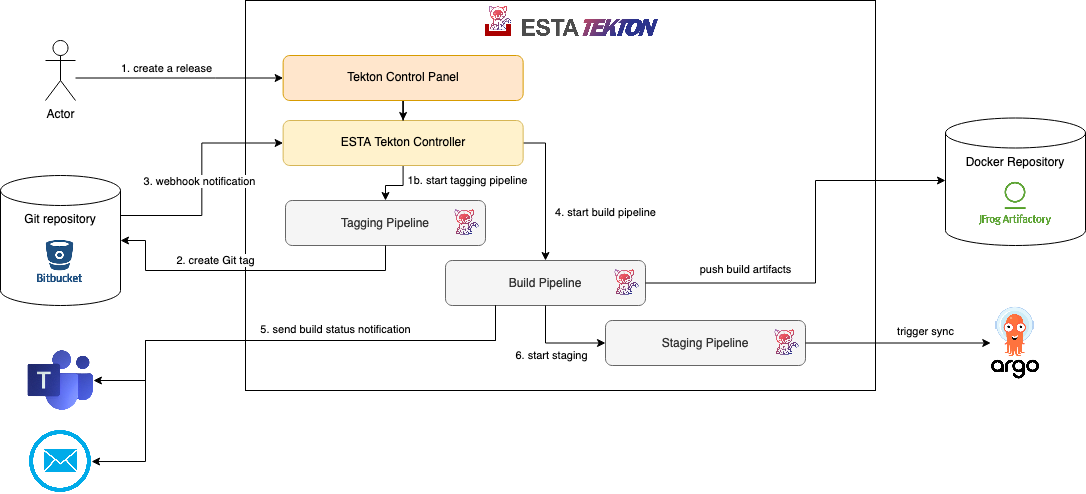
rollme: {{ randAlphaNum 8 | quote }}

{{- end }}

# Release Process

ESTA Tekton triggers the release build by a git tag. The GIT Tag is created by, developers or release managers directly through Bitbucket. It is also possible to create a new tag with the Tagging Assistant in Tekton Control Panel. Using the tagging assistant, the tagging pipeline not only creates a Git tag but also sets the given version to meta files like pom.xml, package.json and to Helm charts found inside the repository.

Another option is to start the tagging process over the Esta Tekton REST service. All actions which can be executed with the Esta Tekton Controller UI can also be executed through REST API calls.



1. Create a release via the Tekton Control Panel
2. Tagging pipeline sets new version and creates a tag in Git
3. Bitbucket webhook triggers release build in ESTA Tekton Controller
4. Build pipeline runs, executes tests and creates build artifacts (Docker image, JAR, Helm charts)
5. Tekton reports build status to configured recipients
6. Staging pipeline updates Helm charts (optional) and triggers ArgoCD to sync

## Auto-releasing from Git Commits

Instead of starting the release process via the Tekton Control Panel or the REST API, a Tagging Pipeline can also be triggered by a Git commit on a certain branch (e.g. on pull request merge into master). With an according pipeline config block in the estaTektonPipeline.json, the parameters for the tagging pipeline (e.g. the automatic increment of the version) are defined. The pipeline trigger criteria like triggerType and branchNamePrefixes are evaluated to start a tagging pipeline on the desired events.

**Tagging pipeline configuration example** Quelle erweitern

{

...

"pipelines": [

...

{

"name": "continuous-release",

"triggerType": [

"GITEVENT"

],

"branchNamePrefixes": [

"master"

],

"tagging": {

"versionIncrementPosition": "none",

"preReleaseVersionIncrement": "patch"

}

}

]

}

This example will start a tagging pipeline on each commit to the master branch which simply removes the -SNAPSHOT suffix from the pom.xml version ("versionIncrementPosition": "none") to create the release version and then set the new -SNAPSHOT version by incrementing the patch position.

See the [Tagging pipeline parameter documentation](file:///C:\display\CLEW\ESTA+Tekton+Parameters+-+estaTektonPipeline.json#ESTATektonParametersestaTektonPipeline.json-Taggingpipelineconfigurationstructure) for reference.

Control increment position with commit message

A common use-case is to decide about the severity of a change at the time when a change is committed to Git or a PR is merged or. With specific keywords inside the commit message one can override the configured versionIncrementPosition just for this change. Use one of the following terms to define whether the automatic release should be a major, minor or patch version increment: #major, #minor, #patch.

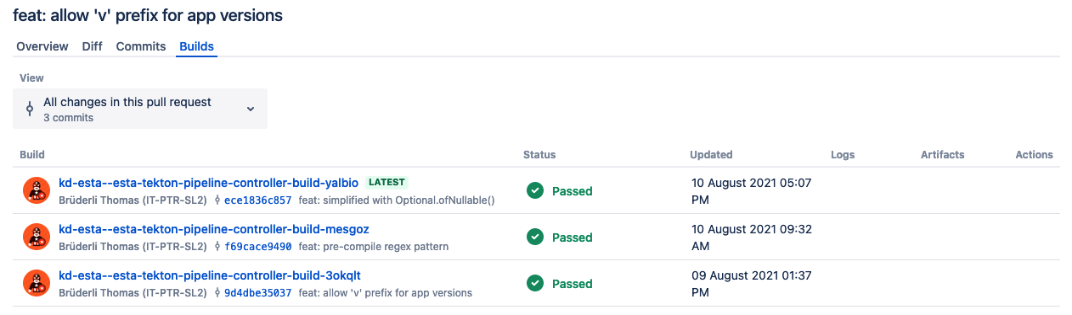
# Bitbucket Integration

## Project Scanner

The ESTA Tekton Controller periodically scans the configured Bitbucket projects for repositories which have a estaTektonPipeline.json (or .yaml) file in the root directory of their default branch. If such a file is detected, the repository is considered "tekton-enabled" and listed in the Tekton Control Panel. At the same time, the ESTA Tekton Controller registers a webhook to receive notifications on Git events from that repository.

## Build Status Notifications

The status of build pipeline runs (started, success, failure) is immediately reported to Bitbucket using a reference to the commit/revision used in that pipeline run. The build status is then visualized in Bitbucket and can even be used as a hard criteria to merge pull requests.



# Notifications

Notifications for pipeline runs and their termination status can be configured individually for each repository via the estaTektonPipeline.json file. Tekton can currently send notifications via email or Microsoft Teams. By default, build notifications with the termination status (success, failure) will be sent as email to the actor (i.e. the user who started the pipeline via Git push or Tekton Control Panel). Here's an email and teams notifications exmaple config:

**estaTektonPipeline.json Notificatons Example** Quelle erweitern

{

...

"notifications": [

{

"type": "EMAIL",

"recipients": ["$actor", "some.one-else@sbb.ch"],

"eventTypes": ["SUCCESS", "FAILURE"],

"branchNameExcludes": ["develop"]

   },

{

"type": "TEAMS",

"enabled": false,

"webhookSecretName": "<secret-holding-connector-webhook-url>",

"pipelineTypes": ["BUILD","STAGING"],

"eventTypes": ["SUCCESS", "FAILURE", "ABORTED"],

"title": "${PIPELINE\_TYPE} Pipeline has ${PIPELINE\_STATUS}"

}

]

}

You can specify any number of notification profiles with different type, filters and recipients.

See the [Parameter Documentation](file:///C:\display\CLEW\ESTA+Tekton+Parameters+-+estaTektonPipeline.json#ESTATektonParametersestaTektonPipeline.json-Notificationconfigurationstructure) for a complete description of all notification parameters.

## Variable substitution

Notification texts and titles can be parametrized with placeholder values in the form ${VAR\_NAME}. The following variables can be used to compose notification messages:

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

**PIPELINE\_TYPE** - The pipeline type that triggered this notification (one of "TAGGING", "BUILD", "STAGING", "CUSTOM")

**PIPELINE\_STATUS** - The termination status of the pipeline (one of "SUCCESS", "FAILED", "ABORTED")

**PIPELINE\_CONFIG\_NAME** - Name of the pipeline config profile used

**PIPELINERUN\_NAME** - Unique name of the pipeline run

**PIPELINERUN\_URL** - URL to the pipeline run details page in the Tekton Control Panel

**PIPELINERUN\_CONSOLE\_URL** - URL to the pipeline run view in the OpenShift Console

**GIT\_REVISION** - Git commit hash used in the pipeline run

**GIT\_BRANCH** - Git branch the pipeline was started from (not always present)

**GIT\_URL** - URL to the Git repository that was cloned into the pipeline workspace

**VERSION\_TAG** - Release version (for tagging pipelines) or Git tag (for build and staging pipelines)

**ERROR** - Error message for failed pipeline runs

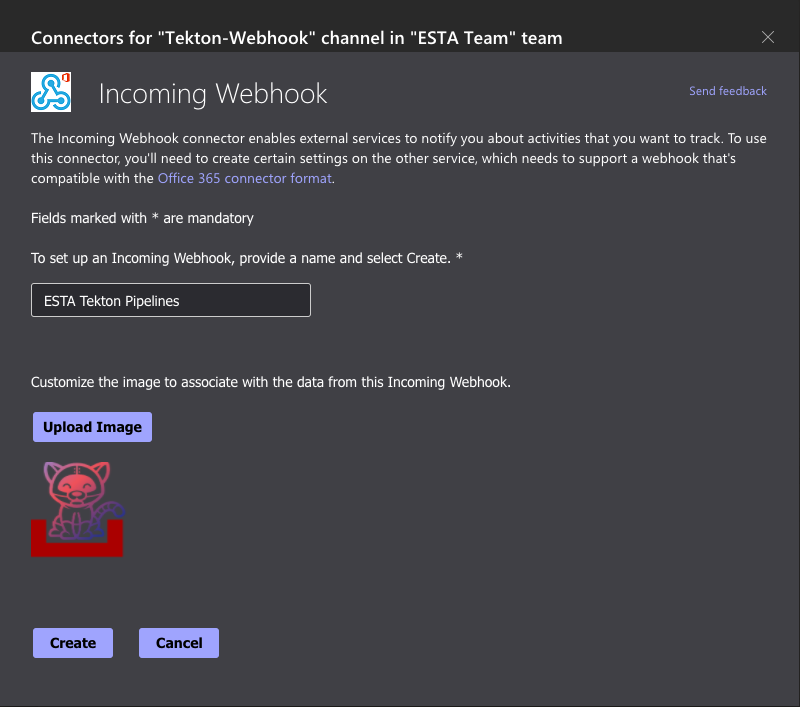
## Email Notifications

For receiving email notifications, no specific configuration is needed. List all recipients to receive a notification message. The value "$actor" will be replaced with the email address of the initiating user.

## MS Teams Notifications

In order to receive notifications in MS Teams, a webhook URL needs to be entered along with the general filters and the "type":"TEAMS" property. Webhook URLs can be registered to channels in MS Teams:

1. Choose "Connectors" from the Options menu of the desired channel in MS Teams
2. Search for the "Incoming Webhook" connector and click the "Add" button
3. Give it a name and optionally upload a nice icon
4. Click the "Create" button
5. Copy the URL shown in the dialog and save it to a [Pipeline Secret](file:///C:\display\CLEW\Tekton+Control+Panel#TektonControlPanel-PipelineSecrets) with the key "URL"
6. Configure the secret name in your estaTektonPipeline.json file



## Customize Default Notification Texts

The default notification texts can be configured for the entire ESTA Tekton build namespace. [Talk to us](https://flow.sbb.ch/servicedesk/customer/portal/8/create/34) if you want to customize them.