

Content Page

1 Introduction	2
1.1 Overall process	2
1.2 Terminology	3
1.3 Developer assets	3
2 Pushing	5
2.1 Pushing entire project	5
2.2 Pushing single asset	7
2.3 SnapLogic Pipelines involved	8
3 Pulling	9
3.1 Pulling entire project	10
3.2 Pulling single asset	11
3.3 SnapLogic Pipelines involved	13
4 Bitbucket CI/CD configuration	14
4.1 CI/CD assets	15
4.2 SnapLogic Pipelines involved	16
5 Full Commit and Reviewing process	16
5.1 Committing the new changes	17
5.2 Creating Pull Request	20
5.3 Pipeline comparison	21
5.4 Approving and merging Pull Request	24

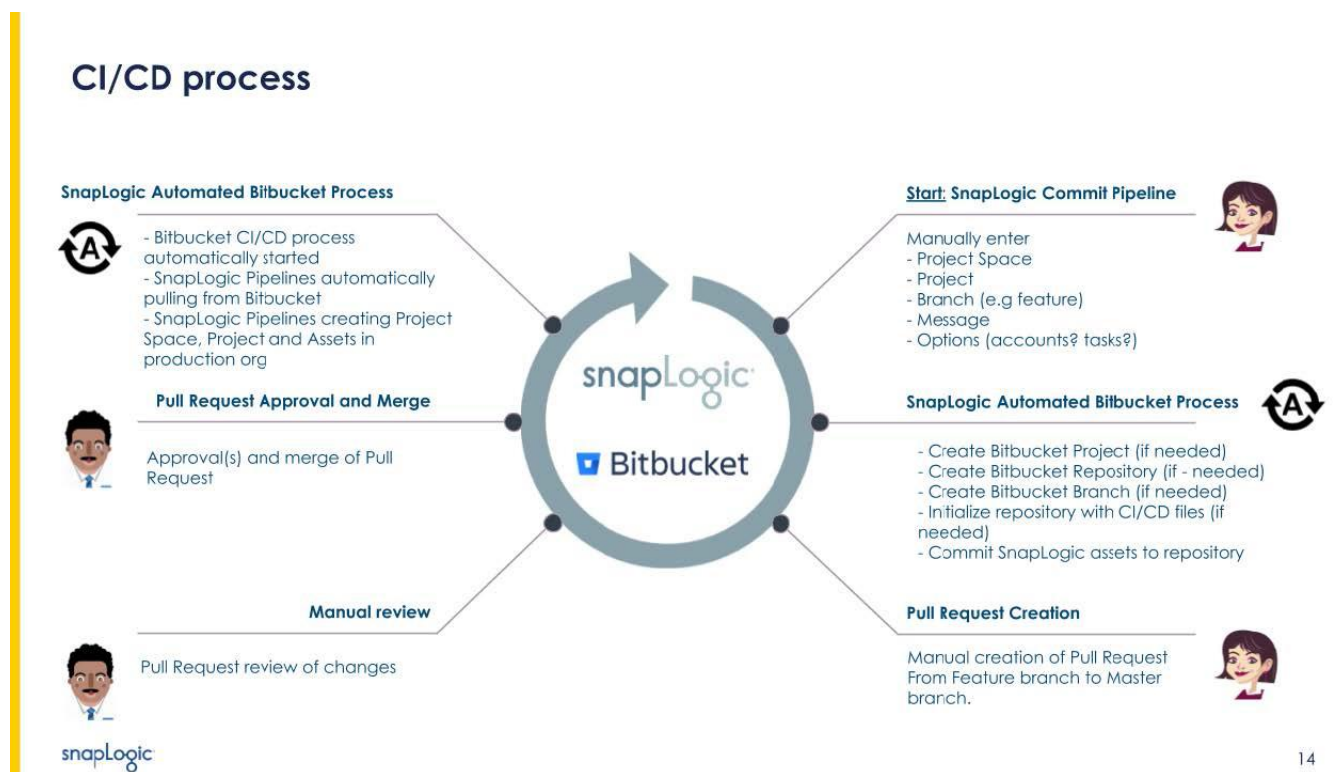
1 Introduction

A video showing a full demonstration of the CI/CD solution can be seen here:

<https://snaplogic.box.com/s/y5fimpt77j2yqpwcqwxpbnfwhhars5b>

1.1 Overall process

This documentation aims to cover the entire proposed CI/CD process outlined in the image below.



The implementation and documentation will enable the following capabilities

- Ability to source control any SnapLogic Pipeline, Task and Account
 - Commit entire project
 - Commit individual asset
 - Specify commit message
 - Specify branch name
- Automatic Bitbucket project, repository and branch creation
- Automatic Bitbucket CI/CD file upload and Pipeline enablement
- Automatic SnapLogic project space, project and asset creation

- Ability to pull assets from Bitbucket to a SnapLogic project
 - Revert changes based on specific branch
 - Revert entire project or specific asset
- SnapLogic Compare Pipeline review
- Bitbucket Pull Request creation, approval and merge
- Automatic promotion of assets from development to production

1.2 Terminology

The relationship between SnapLogic organizations, project spaces projects and Bitbucket projects, repositories and branches are as follows.

- A SnapLogic project space (belongs to a SnapLogic organization) will be mapped to a Bitbucket repository
- A SnapLogic project (belongs to a SnapLogic project space) will be mapped to a directory/folder within that repository
- Each repository will have 1 or more Bitbucket branches. By default, the master branch will reflect the state of assets in the SnapLogic production organization. Additional branches (feature branches) inherits the master branch and will reflect various new development efforts in the SnapLogic development organization

1.3 Developer assets

Each SnapLogic user that should be involved in committing or pulling assets to the Bitbucket space could have its unique and individual assets. The template of the assets and tools involved in the process can be found under the *Company_dev/Administration/User_Bitbucket* project.

It is recommended that each user duplicates (Export -> Import) the *User_Bitbucket* project and replaces User with its unique name.

For impersonating the individual user on Bitbucket (e.g for author of commits), the Account used in the SnapLogic Pipelines need to refer to the user's individual Bitbucket Account. For each Pipeline contained in the *User_Bitbucket* project, the single Snap holds a Pipeline Parameter called *bitbucket_account*. This needs to be replaced for each user's copy of the *User_Bitbucket* project, to reflect the path to the individual's own Bitbucket Account.

To create a new Bitbucket Account, follow these steps

1. Create a Bitbucket OAuth as instructed here <https://support.atlassian.com/bitbucket-cloud/docs/use-oauth-on-bitbucket-cloud/#Create-a-consumer>

The following *Callback URL* setting should be used:

<https://elastic.snaplogic.com/api/1/rest/admin/oauth2callback/rest>

The below Permissions should be applied

Permissions

Account	<input type="checkbox"/> Email	Pull requests	<input checked="" type="checkbox"/> Read
	<input type="checkbox"/> Read		<input checked="" type="checkbox"/> Write
	<input type="checkbox"/> Write	Issues	<input checked="" type="checkbox"/> Read
Workspace membership	<input type="checkbox"/> Read		<input checked="" type="checkbox"/> Write
	<input type="checkbox"/> Write	Wikis	<input type="checkbox"/> Read and write
Projects	<input checked="" type="checkbox"/> Read	Snippets	<input type="checkbox"/> Read
	<input checked="" type="checkbox"/> Write		<input type="checkbox"/> Write
Repositories	<input checked="" type="checkbox"/> Read	Webhooks	<input checked="" type="checkbox"/> Read and write
	<input checked="" type="checkbox"/> Write	Pipelines	<input checked="" type="checkbox"/> Read
	<input checked="" type="checkbox"/> Admin		<input checked="" type="checkbox"/> Write
	<input checked="" type="checkbox"/> Delete		<input checked="" type="checkbox"/> Edit variables

2. Copy the *key* and *secret* of the created consumer
3. In SnapLogic, create a *REST OAuth2 Account* under the user's individual project that was created earlier.

For *Client ID*, paste the *key*

For *Client secret*, paste the *secret*

Check *Header authentication*

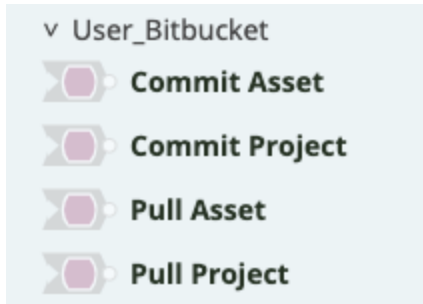
For *OAuth2 Endpoint*, enter <https://bitbucket.org/site/oauth2/authorize>

For *OAuth2 Token*, enter https://bitbucket.org/site/oauth2/access_token

For *Grant Type*, select *authorization_code*

4. Press *Authorize* and wait for the token to be retrieved
5. *Apply* and save the Account

Although covered in greater detail in the upcoming sections, the *User_Bitbucket* project holds these four Pipelines, each containing a single Snap



1. Commit Project - Commits any Pipelines, Accounts and Tasks within the specified SnapLogic project, to the specified branch in Bitbucket
2. Commit Asset - Commits the specified asset within the specified SnapLogic project, to the specified branch in Bitbucket
3. Pull Project - Reads any Pipelines, Accounts and Tasks from the specified branch in the specified Bitbucket, to the specified project and organization of SnapLogic
4. Pull Asset - Reads the specified asset from the specified branch in the specified Bitbucket, to the specified project and organization of SnapLogic

For each Pipeline, each user needs to update the *bitbucket_account* Pipeline Parameter in the respective Snaps, matching the path to their own Bitbucket Account.

2 Pushing

Pushing SnapLogic Pipelines to Bitbucket essentially involves two steps

1. Creation of Bitbucket project and Bitbucket repository, as well as commit of CI/CD assets on master branch. This first step only happens if this was the first time an asset was pushed for the particular SnapLogic project. The CI/CD assets pushed in this step is covered in greater detail in the *Bitbucket CI/CD configuration* section.
2. Read asset from SnapLogic and push to Bitbucket

2.1 Pushing entire project

The *Commit Project* Pipeline in the user's *User_Bitbucket* project is responsible for triggering the process of pushing the assets for an entire project to Bitbucket. It consists of a single Pipeline Execute Snap



Commit Project ? [Icons]

Settings Views Info

Label*
Commit Project

Pipeline*
= ../CICD-BitBucket/1.0 Main - SL Project to Bitbucket

Execute On
SNAPLEX_WITH_PATH

Snaplex Path
=

Execution Label
=

Pipeline Parameters + -

Parameter Name	Parameter Value
workspace	=
source_org	= dev
source_space	= ExampleProjectSpace
source_proj	= ExampleProject
branch	= feature
bitbucket_account	= ../User_Bitbucket/UserBitb
include_task	= true
include_account	= true
message	= "All assets"

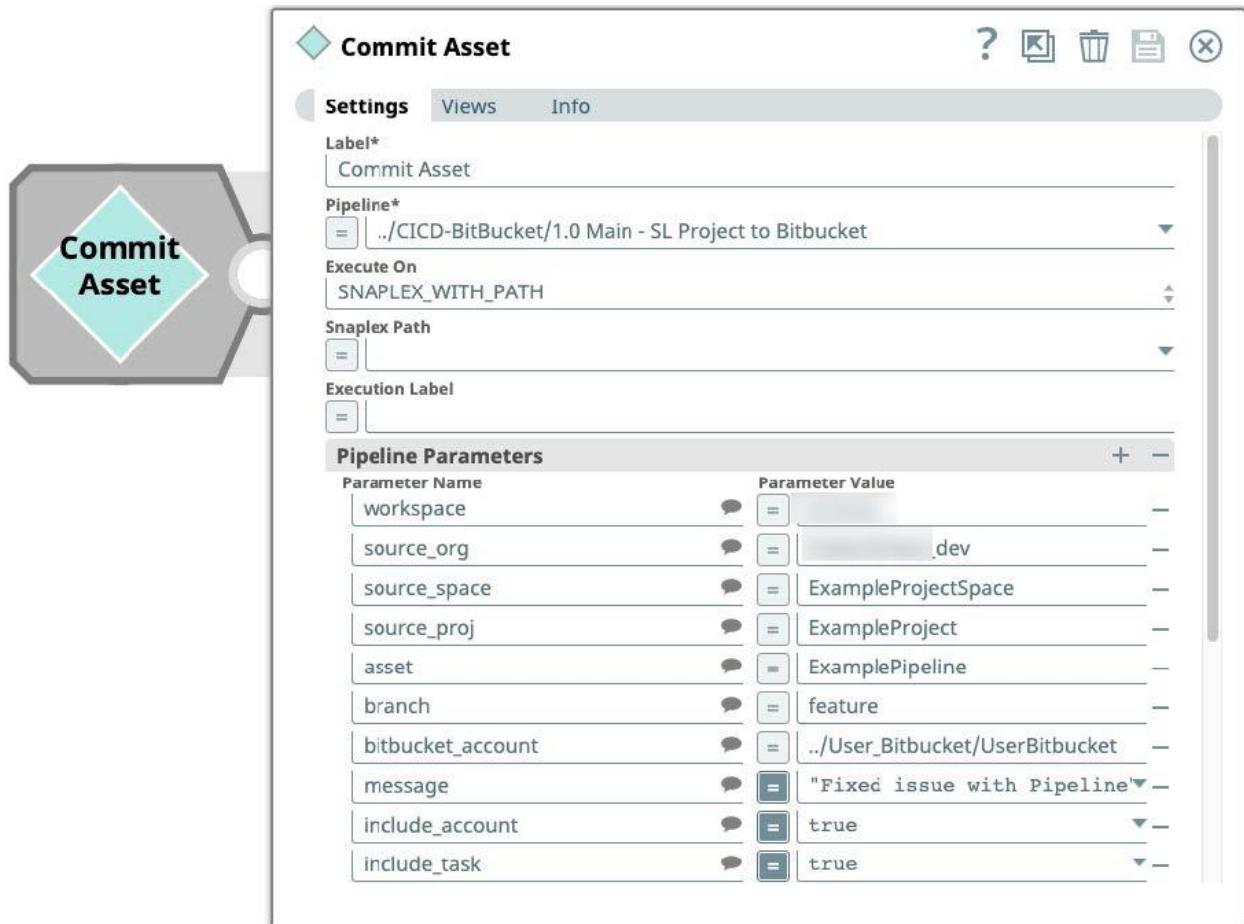
The following parameters are supported

- **workspace** - the name of the Bitbucket workspace ID/slug. For example *companyconnect*
- **source_org** - the name of the SnapLogic organization for where to find the assets to push. For example *CompanyGlobal_dev*
- **source_space** - the name of the SnapLogic project space in where the project is located. For example *ExampleProjectSpace*
- **source_proj** - the name of the SnapLogic project in where the assets are located. For example *ExampleProject*
- **branch** - the name of a Bitbucket branch. The branch will exist in the repository called "ExampleProject" (name of the source_proj) which sits in the "ExampleProjectSpace" (name of the source_space). If the branch does not already exist, it will be created automatically. For example *feature* or *myNewFeature*
- **bitbucket_account** - the relative path to the user's Bitbucket Account (REST OAuth2 Account). This will be the author of any commits made. For example, *../User_Bitbucket/UserBitbucket*.

- **include_task** - whether or not to push SnapLogic Tasks to the repository
- **include_account** - whether or not to push SnapLogic Accounts to the repository
- **message** - an optional commit message that will be reflected in Bitbucket. For example *All assets*

2.2 Pushing single asset

The *Commit Asset* Pipeline in the user's *User_Bitbucket* project is responsible for triggering the process of pushing a single asset within a project to Bitbucket. It consists of a single Pipeline Execute Snap



Commit Asset

Settings Views Info

Label*
Commit Asset

Pipeline*
= ../CICD-BitBucket/1.0 Main - SL Project to Bitbucket

Execute On
SNAPLEX_WITH_PATH

Snaplex Path
=

Execution Label
=

Pipeline Parameters + -

Parameter Name	Parameter Value
workspace	= [redacted]
source_org	= [redacted].dev
source_space	= ExampleProjectSpace
source_proj	= ExampleProject
asset	= ExamplePipeline
branch	= feature
bitbucket_account	= ../User_Bitbucket/UserBitbucket
message	= "Fixed issue with Pipeline"
include_account	= true
include_task	= true

The following parameters are supported

- **workspace** - the name of the Bitbucket workspace ID/slug. For example *companyconnect*
- **source_org** - the name of the SnapLogic organization for where to find the assets to push. For example *CompanyGlobal_dev*

- **source_space** - the name of the SnapLogic project space in where the project is located. For example ExampleProjectSpace
- **source_proj** - the name of the SnapLogic project in where the assets are located. For example ExampleProject
- **asset** - the name of the SnapLogic asset (Pipeline, Task or Account) within the above specified project. This will be the single and only asset pushed to Bitbucket. For example ExamplePipeline
- **branch** - the name of a Bitbucket branch. The branch will exist in the repository called "ExampleProject" (name of the source_proj) which sits in the "ExampleProjectSpace" (name of the source_space). If the branch does not already exist, it will be created automatically. For example *feature* or *myNewFeature*
- **bitbucket_account** - the relative path to the user's Bitbucket Account (REST OAuth2 Account). This will be the author of any commits made. For example, *../User_Bitbucket/UserBitbucket*.
- **include_task** - whether or not to push SnapLogic Tasks to the repository
- **include_account** - whether or not to push SnapLogic Accounts to the repository
- **message** - an optional commit message that will be reflected in Bitbucket. For example *Fixed issue with Pipeline*

2.3 SnapLogic Pipelines involved

- **1.0 Main - SL Project to Bitbucket.** Both Pipelines used above take advantage of the main Pipeline that sits in *CompanyGlobal_dev/Administration/CICD-BitBucket*. The Pipeline can be found here: https://elastic.snaplogic.com/sl/designer.html?v=c5d80f#pipe_snode=6001b78a2c05498aaefc3e34
 - Immediately, the **1.1 Create Project and Repo** Pipeline is called
 - Router then checks whether the user wanted to include Tasks and Accounts
 - Each route will list all SnapLogic assets within the specified SnapLogic project using the *SnapLogic List Snaps*.
 - For each listed asset, the name of the asset is checked against a potential Pipeline Parameter value for a single *asset*. This is to exclude any non matched assets if the Pipeline was started from the *Commit Asset* Pipeline, intended to only commit a single asset.
 - A *Conditional Snap* is used to determine the type of each asset, e.g Pipeline, Task or Account.
 - For each included asset, the **1.2 SL Asset to Bitbucket** Pipeline is called
- **1.1 Create Project and Repo.** The purpose of this Pipeline is to create a Bitbucket project and repository, as well as adding CI/CD files and initializing the Bitbucket CI/CD Pipeline feature. This Pipeline can be found here: https://elastic.snaplogic.com/sl/designer.html?v=c5d80f#pipe_snode=5ffd7839a3c9d3c75891559a
 - Calls the Bitbucket API (<https://developer.atlassian.com/bitbucket/api/2/reference/resource/workspaces/>)

[%7Bworkspace%7D/projects#post](#)) to create a new Bitbucket project, mapping to the SnapLogic project space. In the Mapper, a Bitbucket project Key is created from the first four characters of the project space name. If this call fails, it is assumed that the project already exists.

- Calls the Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D#post) to create a new Bitbucket repository, mapping to the SnapLogic project. If this call fails, it is assumed that the repository already exists
- If a new repository was created, it will call the Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/src#post) to upload two files in a new commit. The two files will be used for Bitbucket CI/CD and are covered in a later section
- After having uploaded the two files, a new call is made to the Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/pipelines_config) to enable the repository for CI/CD. This is described in more detail here: <https://bitbucket.org/product/features/pipelines>
- **1.2 SL Asset to Bitbucket.** This Pipeline will read the assets from SnapLogic and push them to Bitbucket in commits. This Pipeline can be found here: https://elastic.snaplogic.com/sl/designer.html?v=c5d80f#pipe_snode=5ffd1452e731db408a73498b
 - *Router* routes to a particular *SnapLogic Read Snap*, based on the asset type (e.g Pipeline, Account and Task)
 - *SnapLogic Read Snaps* are used to extract the JSON object of the particular asset.
 - A *Mapper Snap* is used to encode and stringify the JSON object to make it ready for the next Snap
 - The Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/src#post) is then used to create an actual commit with the encoded JSON SnapLogic asset.
Note: if the asset already exists in the repository, it will be overwritten if there has been any changes to the asset.

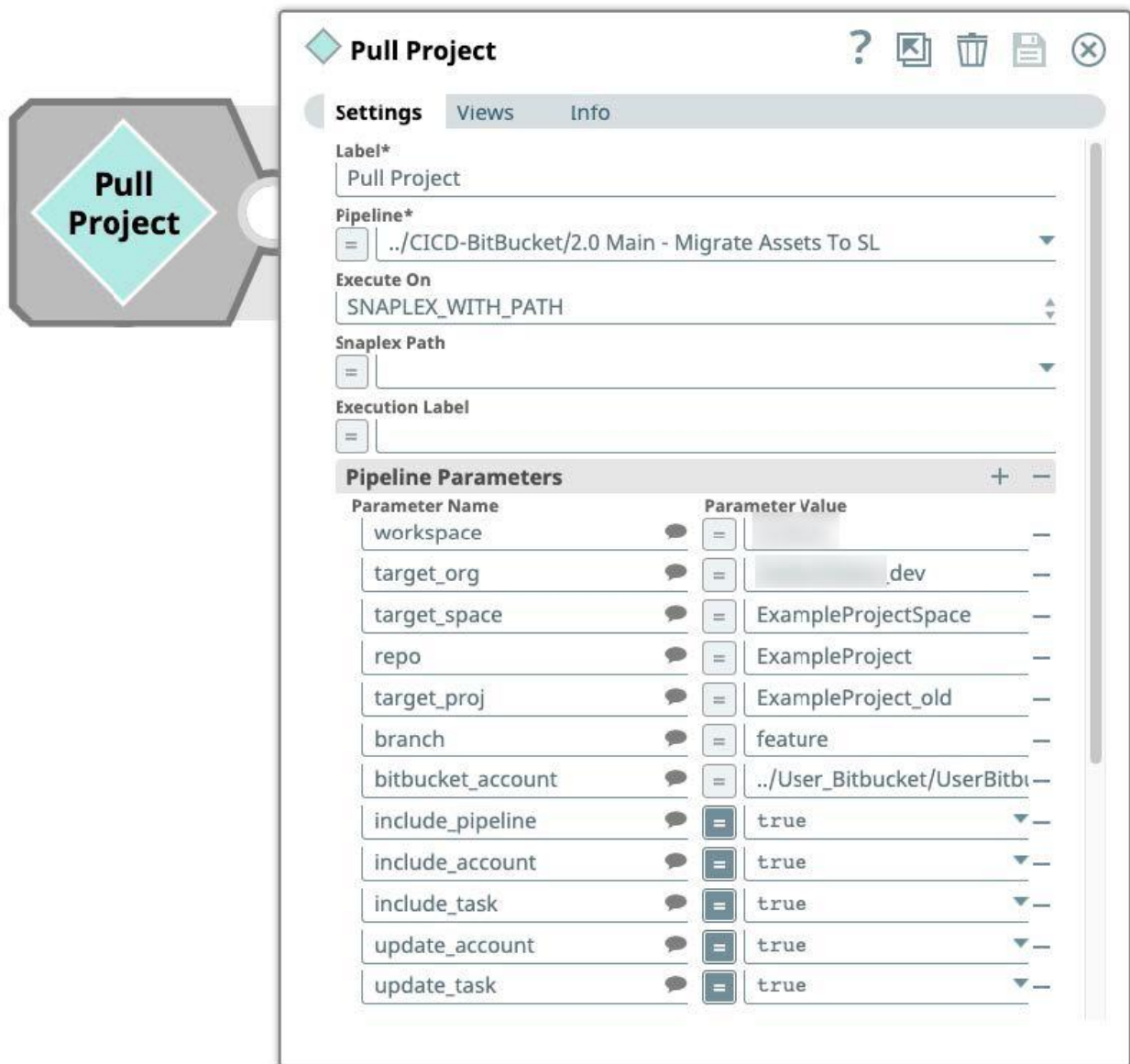
3 Pulling

Pulling refers to the process of reading assets from a Bitbucket repository and then updating or creating them as SnapLogic assets. This can either be done manually by a user, as part of reverting to a previous versioned copy of an asset, or to create a new temporary project where the Pipelines correspond to different versions, so that two Pipelines can be compared using the visual tool. This will be described in more detail under the *Reviewing* section.

Much of the Pipeline logic described in this section will also be used when the CI/CD automated processes promote assets from the development organization to the production organization. This is covered in a later section.

3.1 Pulling entire project

The *Pull Project* Pipeline in the user's *User_Bitbucket* project is responsible for triggering the process of reading the assets from Bitbucket and creating or updating them in SnapLogic. It consists of a single Pipeline Execute Snap



Pull Project

Settings Views Info

Label*
Pull Project

Pipeline*
../CICD-BitBucket/2.0 Main - Migrate Assets To SL

Execute On
SNAPLEX_WITH_PATH

Snaplex Path

Execution Label

Pipeline Parameters + -

Parameter Name	Parameter Value
workspace	
target_org	dev
target_space	ExampleProjectSpace
repo	ExampleProject
target_proj	ExampleProject_old
branch	feature
bitbucket_account	../User_Bitbucket/UserBitb
include_pipeline	true
include_account	true
include_task	true
update_account	true
update_task	true

The following parameters are supported

- **workspace** - the name of the Bitbucket workspace ID/slug. For example *companyconnect*

- **target_org** - the name of the SnapLogic organization for where the assets should be created or updated. For the automated promotion of assets, this will default to *CompanyGlobal_prod*. But for the manual operation covered in this section, it will typically be *CompanyGlobal_dev*.
- **target_space** - the name of the SnapLogic project space in where the project is located. For example *ExampleProjectSpace*
- **repo** - the name of the Bitbucket repository for where the assets are located. This should always map to the SnapLogic project. For example *ExampleProject*
- **target_proj** - the name of the SnapLogic project to where the assets should be created or updated. There are two different use cases for this property
 - When the *target_proj* is set to equal the *repo*, the logic of this operation becomes a revert. It will look at the Bitbucket repository, extract the assets and create and update the assets in the project where the assets originally existed from. This can be done if a user wants to replace an updated or removed asset with what exists on a particular branch in the repository.
 - When the *target_proj* is different to the *repo*, the logic of this operation becomes a copy/mirror. It will look at the Bitbucket repository, extract the assets and create and update the assets in the project that will be used as a reference to compare the new assets. If the project does not exist, it will be automatically created. In most cases, this will be used together with specifying the branch as *master*.
- **branch** - the name of a Bitbucket branch. This must be an existing branch.
- **bitbucket_account** - the relative path to the user's Bitbucket Account (REST OAuth2 Account). For example, *../User_Bitbucket/UserBitbucket*.
- **include_task** - whether or not to pull SnapLogic Tasks from the repository
- **include_account** - whether or not to pull SnapLogic Accounts from the repository
- **include_pipeline** - whether or not to pull SnapLogic Pipelines from the repository
- **update_account** - whether or not to replace the existing version of the asset if it exists already
- **update_task** - whether or not to replace the existing version of the asset if it exists already

3.2 Pulling single asset

The *Pull Asset* Pipeline in the user's *User_Bitbucket* project is responsible for triggering the process of reading a single asset from Bitbucket and creating or updating them in SnapLogic. It consists of a single Pipeline Execute Snap



Pull Asset

Settings Views Info

Label*
Pull Asset

Pipeline*
../CICD-BitBucket/2.0 Main - Migrate Assets To SL

Execute On
SNAPLEX_WITH_PATH

Snaplex Path

Execution Label

Pipeline Parameters

Parameter Name	Parameter Value
workspace	
target_org_dev
target_space	ExampleProjectSpace
repo	ExampleProject
target_proj	ExampleProject_old
asset	ExamplePipeline
branch	feature
bitbucket_account	../User_Bitbucket/UserBitbu

The following parameters are supported

- **workspace** - the name of the Bitbucket workspace ID/slug. For example *companyconnect*
- **target_org** - the name of the SnapLogic organization for where the asset should be created or updated
- **target_space** - the name of the SnapLogic project space in where the project is located. For example *ExampleProjectSpace*
- **repo** - the name of the Bitbucket repository for where the asset is located. This should always map to the SnapLogic project. For example *ExampleProject*
- **target_proj** - the name of the SnapLogic project to where the asset should be created or updated. There are two different use cases for this property
 - When the *target_proj* is set to equal the *repo*, the logic of this operation becomes a revert. It will look at the Bitbucket repository, extract the asset and create and update the asset in the project where the asset originally existed from. This can be done if a user wants to replace an updated or removed asset with what exists on a particular branch in the repository.
 - When the *target_proj* is different to the *repo*, the logic of this operation becomes a copy/mirror. It will look at the Bitbucket repository, extract the asset and create

and update the asset in the project that will be used as a reference to compare the new asset. If the project does not exist, it will be automatically created. In most cases, this will be used together with specifying the branch as *master*.

- **branch** - the name of a Bitbucket branch. This must be an existing branch.
- **bitbucket_account** - the relative path to the user's Bitbucket Account (REST OAuth2 Account). For example, *../User_Bitbucket/UserBitbucket*.
- **include_task** - whether or not to pull SnapLogic Tasks from the repository
- **include_account** - whether or not to pull SnapLogic Accounts from the repository
- **include_pipeline** - whether or not to pull SnapLogic Pipelines from the repository
- **update_account** - whether or not to replace the existing version of the asset if it exists already
- **update_task** - whether or not to replace the existing version of the asset if it exists already

3.3 SnapLogic Pipelines involved

- **2.0 Main - Migrate Assets To SL.** Both Pipelines used above take advantage of the main Pipeline that sits in *CompanyGlobal_dev/Administration/CICD-BitBucket*. The Pipeline can be found here:
https://elastic.snaplogic.com/sl/designer.html?v=c5d80f#pipe_snode=5ffd1456a66cd47fc35b7935
 - Immediately, the **2.1 Upsert Space And Project** Pipeline is called
 - Router checks whether the user wanted to include Pipelines, Tasks and Accounts
 - Each route will call the **2.2 Read Assets Pipeline**, passing the type of asset.
- **2.1 Upsert Space And Project.** The purpose of this Pipeline is to create the SnapLogic project space and project. The Pipeline can be found here:
https://elastic.snaplogic.com/sl/designer.html?v=c5d80f#pipe_snode=5ffd145bfbb01ae54875676c
 - Lists all SnapLogic project spaces using the *SnapLogic List Snap*
 - Searches for a match
 - If no match found, creates a new project space using the *SnapLogic Create Snap*
 - Lists all SnapLogic projects using the *SnapLogic List Snap*
 - Searches for a match
 - If no match found, creates a new project using the *SnapLogic Create Snap*
- **2.2 Read Assets.** This Pipeline will find and list all the assets from Bitbucket for the specified asset type. The Pipeline can be found here:
https://elastic.snaplogic.com/sl/designer.html?v=c5d80f#pipe_snode=5ffd1453d9ea7bc620a996ba
 - The Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/refs/branches) is used to retrieve a list of all open branches for the specified repository.
 - A *Filter Snap* is used to exclude all but the specified branch from the list

- A hash value is extracted from the branch response. In Bitbucket, this hash is required to be used to query for assets within a repository.
- For the matched branch, the Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/src/%7Bcommit%7D/%7Bpath%7D) is used to retrieve a list of all files for the specified branch (hash value) for the specified asset type
- As it's not guaranteed there actually exists assets for each asset type in the branch of the repository, an Error View is used to ignore any 404 Not Found responses from the Bitbucket API
- Each asset in the repository is split by a *JSON Splitter* and a hash is extracted from each asset.
- A *Filter Snap* is used to only match a specific asset, if the user had requested a Pull Asset operation.
- A *Router Snap* is used to direct the file to an individual Pipeline, each handling its own type of assets
- **2.2.1 Upsert Pipeline To SL, 2.2.2 Upsert Account To SL and 2.2.3 Upsert Task To SL.** These three Pipelines are very similar, with the main difference being the Asset Type between the Metadata Snap Packs. The Pipelines can be found here: https://elastic.snaplogic.com/sl/designer.html?v=168e8a#pipe_snode=5ffd1457e76eadc405d0f2e3, https://elastic.snaplogic.com/sl/designer.html?v=168e8a#pipe_snode=5ffd145bcfb5a379e64bbc50 and https://elastic.snaplogic.com/sl/designer.html?v=168e8a#pipe_snode=5ffd14544097658bb14647bb
 - The Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/repositories/%7Bworkspace%7D/%7Brepo_slug%7D/src/%7Bcommit%7D/%7Bpath%7D) is called to retrieve the file content of the particular asset for a specific asset type
 - The retrieved file name is joined with the list of existing assets for the asset type, using a *SnapLogic List Snap* and a *Join Snap*.
 - Based on a match or not, a Router looks at the user's update parameter to determine whether to skip, create or update the asset in the target project
 - *SnapLogic Create* or *Update Snaps* are used to potentially update or create the asset in the target project

4 Bitbucket CI/CD configuration

Bitbucket provides a feature called Pipelines, used to deliver CI/CD processes. With this feature, Bitbucket will pick up any successful Pull Request on the *master* branch. The Bitbucket Pipeline will invoke a script that calls a SnapLogic Pipeline exposed as a Task. The idea of the SnapLogic Pipeline is to promote the assets relevant for the Pull Request to the production organization.

4.1 CI/CD assets

As mentioned in [section 2.3](#), the *1.1 Create Project and Repo* Pipeline creates the Bitbucket project and repository as well as pushes two files to the *master* branch, and finally enables the Bitbucket Pipeline feature.

Both files reside under *CompanyGlobal_dev/Administration/CI/CD-BitBucket* and are described below

- **bitbucket-pipelines.yml** - Bitbucket specific descriptor that holds the relevant information for the Bitbucket Pipelines to know when and how to react to changes.

```
image: atlassian/default-image:2

pipelines:
  branches:
    master:
      - step:
          name: 'Promotion'
          script:
            - echo "Starting SnapLogic promotion"
            - chmod +x ./runPromotion.sh
            - ./runPromotion.sh
```

The above configuration tells Bitbucket to execute a step called *Promotion* on changes on the branch *master*. Specifically, the script *runPromotion.sh* should be executed.

- **runPromotion.sh** - this script calls the Pipeline *PromotionRequest* through a Triggered Task called *PromoteAssets*.

```
#!/bin/bash
result=$(curl -s
'https://elastic.snaplogic.com:443/api/1/rest/slsched/feed/
_dev/Administration/CI/CD-BitBucket/PromoteAssets'
-H "WORKSPACE: $BITBUCKET_WORKSPACE" -H "PROJECTKEY: $BITBUCKET_PROJECT_KEY"
-H "REPO: $BITBUCKET_REPO_SLUG" -H "Authorization: Bearer ")
exit 0
```

It passes the following Bitbucket specific variables as HTTP headers (Pipeline Parameters)

- WORKSPACE: The name of the Bitbucket workspace where the Pull Request happened, for example *companyconnect*.
- PROJECTKEY: The four character Key of the Bitbucket project
- REPO: The repository of where the Pull Request happened, e.g *ExampleProject*. This will directly map to a SnapLogic project.

Additionally, an Authorization header is passed to authenticate the request.

To make updates that should be reflected on any new project, update the two files stored under *CompanyGlobal_dev/Administration/CI/CD-BitBucket*. For changes on existing repositories, enter Bitbucket and navigate to the files on the master branch.

NOTE: As any newly created branch will inherit the master branch, every branch will also have the two above files in the root directory. However, they will not do anything as they are restricted to be triggered on the master branch.

4.2 SnapLogic Pipelines involved

- **PromotionRequest.** This Pipeline is invoked through the Triggered Task *PromoteAssets*, under *CompanyGlobal_dev/Administration/CICD-BitBucket*. The Pipeline can be found here:
https://elastic.snaplogic.com/sl/designer.html?v=168e8a#pipe_snode=5ffd8ba1a04bb171a8a7dd15
 - The Bitbucket API (https://developer.atlassian.com/bitbucket/api/2/reference/resource/workspaces/%7Bworkspace%7D/projects/%7Bproject_key%7D) is used to retrieve the full name of the SnapLogic project space, extracted from the four character Key that was passed from the Bitbucket Pipeline (PROJECTKEY)
 - The Bitbucket API (<https://developer.atlassian.com/bitbucket/api/2/reference/resource/workspaces/%7Bworkspace%7D>) is used to retrieve the full name of the SnapLogic project, extracted from the URL-friendly repository name that was passed from the Bitbucket Pipeline (REPO)
 - A Join Snap is used to merge the two returned project space and project names together
 - *Pipeline Execute* Snap is used to call the *2.0 Main - Migrate Assets To SL* Pipeline. That Pipeline is covered under [section 3.3](#). For invoking the Pipeline, *CompanyGlobal_prod* is used as the target SnapLogic organization.

NOTE: The Pipeline Parameter for *bitbucketAccount* should be updated to an Account that is able to read data from all users' repositories.

5 Full Commit and Reviewing process

This section assumes that one or several initial commits have been made already to a new repository, following the information and instructions from a [previous section](#). The purpose of this section is to focus on the Bitbucket specific actions, as well as options for reviewing Pipeline changes manually.

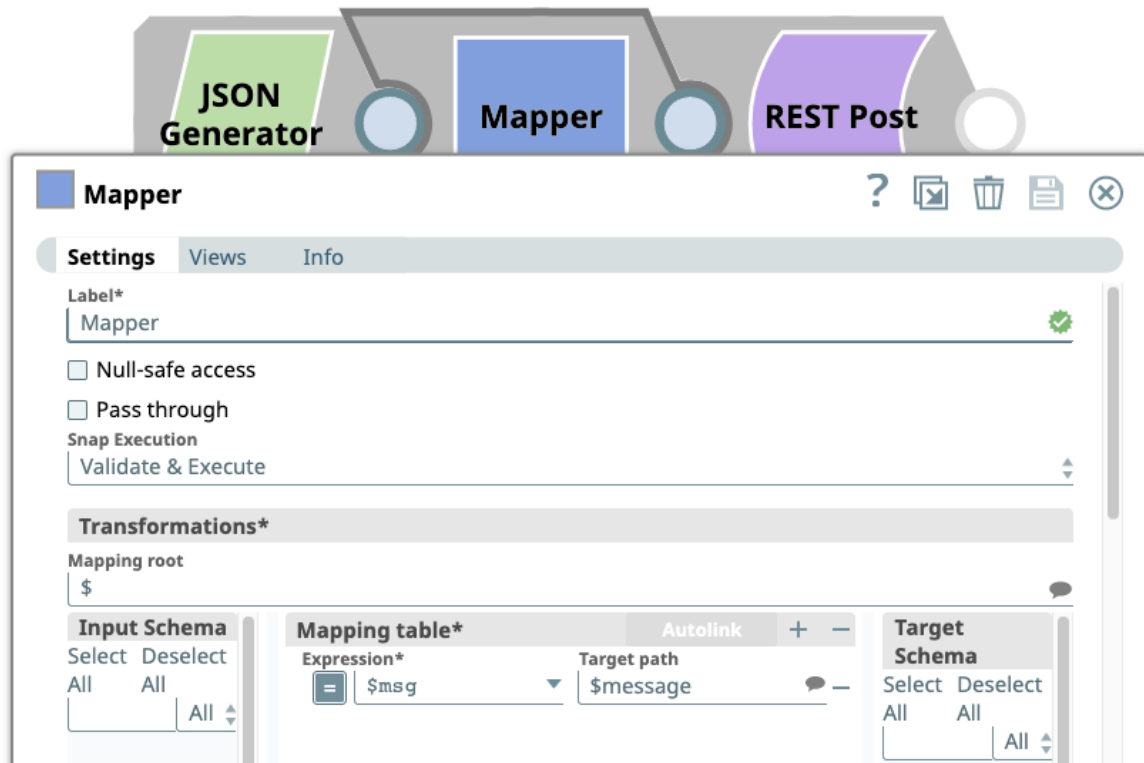
As there would be no comparison opportunity on a fresh new project, we will assume there is already a project running in production, with the relevant assets already existing on the master branch. The task is now to do a few changes in development, pushing them to a new feature branch on Bitbucket, creating a Pull Request, having someone reviewing and merging our changes to master, with Bitbucket finally promoting the assets to production.

Below is the state of assets for our project, with exactly the same assets in both *CompanyGlobal_dev* as well as *CompanyGlobal_prod*

ExampleProjectSpace/ExampleProject

All	Accounts	Files	Tasks	Pipelines	Snap Packs	Snaplexes	Tables
<input type="text" value="Search"/>	<input type="checkbox"/> Exact search						
<input type="checkbox"/>	Name						Type
<input type="checkbox"/>	ExampleTask						Task
<input type="checkbox"/>	ExampleAccount						Account
<input type="checkbox"/>	ExamplePipeline						Pipeline

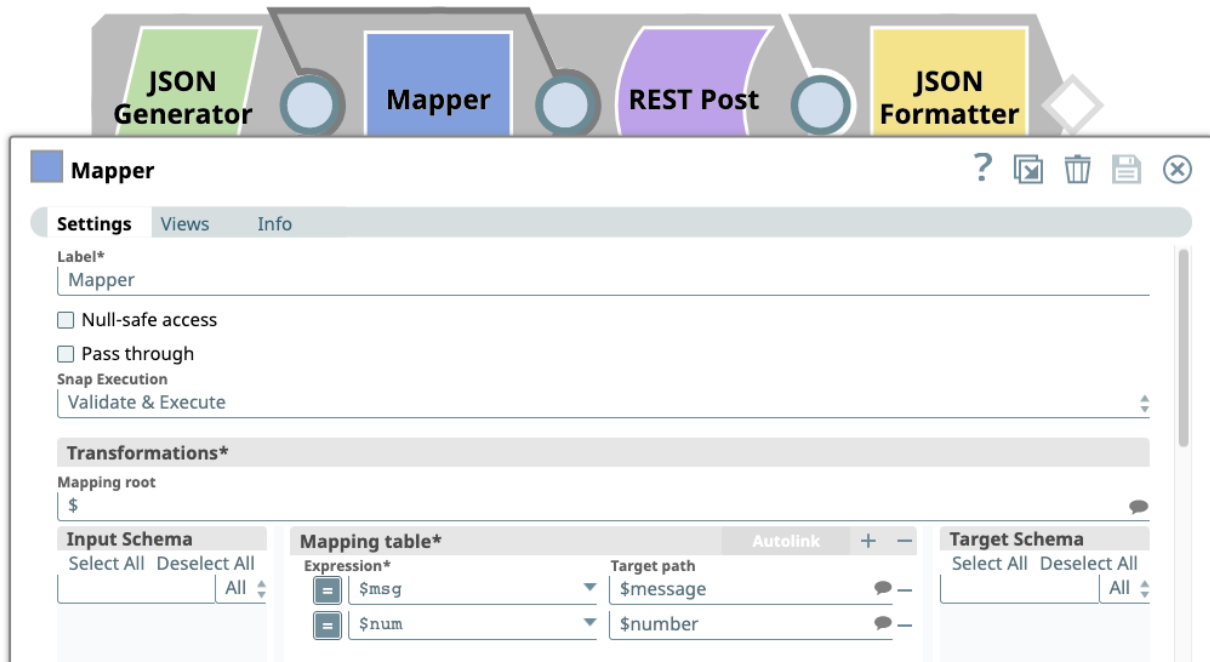
The design of our ExamplePipeline looks like below



It is a Pipeline with three Snaps, including a Mapper Snap with a single Expression in the Mapping table.

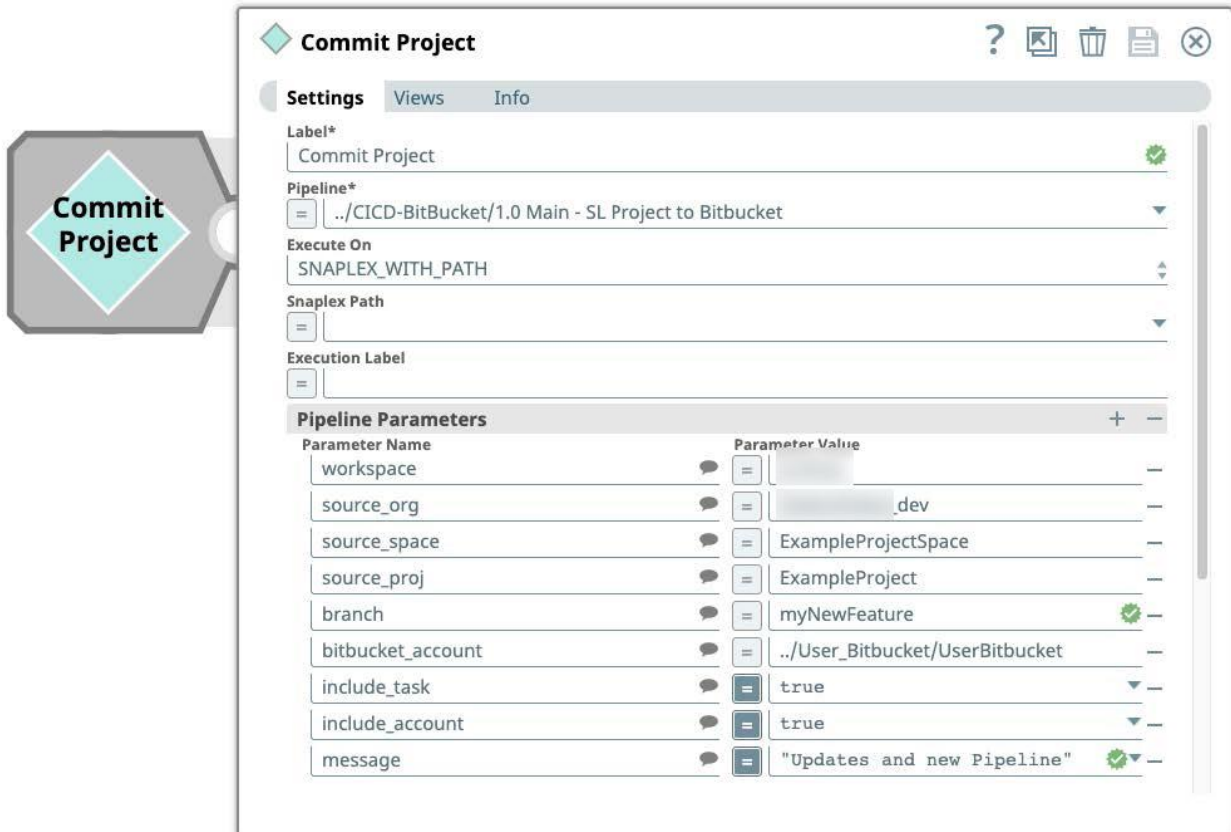
5.1 Committing the new changes

During a new development effort, the Pipeline has been updated, adding a JSON Formatter Snap after the REST Post Snap, as well as adding a second row to the Mapper's Mapping table



Additionally, a new Pipeline has been created, called SecondPipeline. To commit the new assets to Bitbucket, the user has navigated to [its personal Commit Project Pipeline](#) (rather than committing each asset individually with the [Commit Asset Pipeline](#))

The below image shows the settings of the Commit Project Snap in the Pipeline, specifying the location of the SnapLogic project space, project, as well as the name for the new branch. A message has been added to identify the purpose of the commit.



When completed, the Bitbucket repository will look as below. Note that the image is showing a navigated view where the user has entered the *pipelines* folder, as those were the assets updated and created for this particular commit.

Linus Hakansson / ExampleProjectSpace / ExampleProject

pipelines

Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, [add a description to your repository](#).

myNewFeature Files Filter files

exampleproject / pipelines

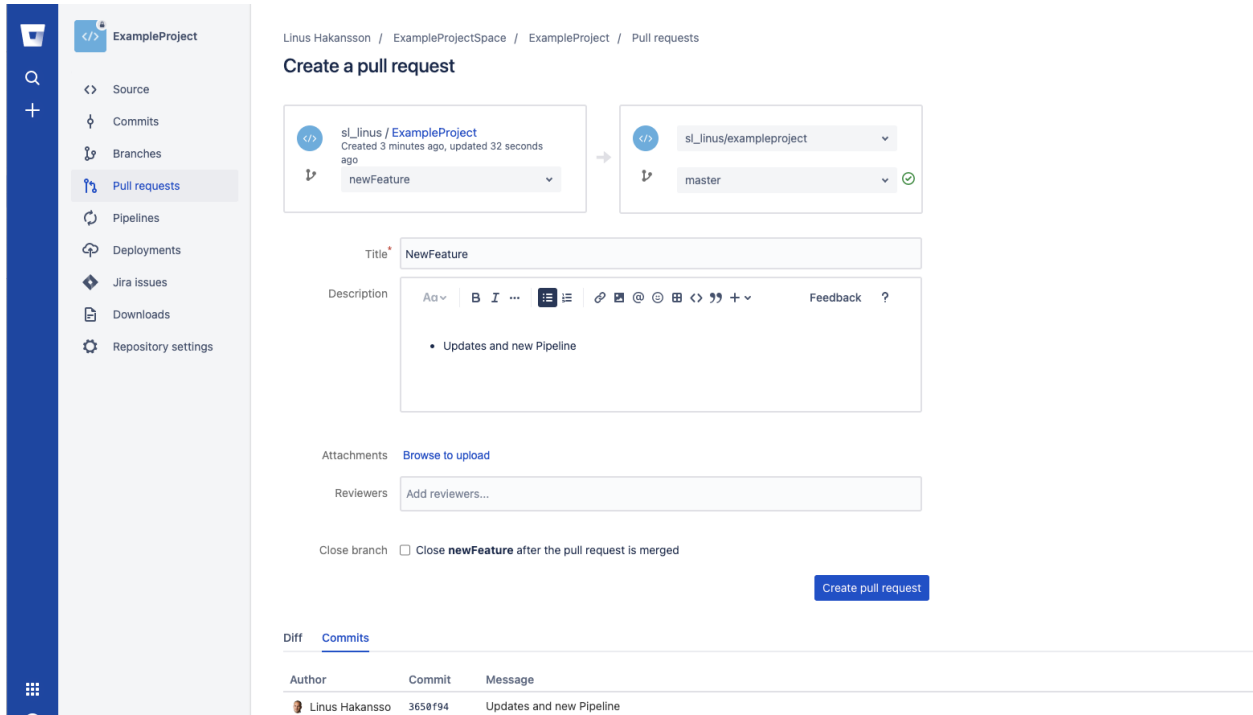
Name	Size	Last commit	Message
..			
ExamplePipeline	14.16 KB	36 seconds ago	Updates and new Pipeline
SecondPipeline	8.15 KB	35 seconds ago	Updates and new Pipeline

As can be seen in the image, the new and updated assets are under the newly automatically created *myNewFeature* branch.

5.2 Creating Pull Request

The next step is to create a Pull Request, with the intention of merging the updates to the master branch (that holds the truth of the CompanyGlobal_prod SnapLogic organization.)

Navigate to the Bitbucket repository's Pull Request Tab and press *Create pull request*.



We can see that three assets have changed between the master branch and our new feature branch.

Diff Commits

Files changed (3)

+1	-1	M	pipelines/ExamplePipeline
+1	-0	A	pipelines/SecondPipeline
+1	-1	M	tasks/ExampleTask

As expected, ExamplePipeline has been modified, and SecondPipeline has been added. Please note that one field has changed on the ExampleTask: the *codegen_url* property. This is a dynamic property. The [1.2 SL Asset to Bitbucket Pipeline](#) could be modified to ignore certain metadata fields for the various asset types.


Press the *Create pull request* button to confirm the defaults.

5.3 Pipeline comparison

With the Pull Request created, colleagues could review the changes by exploring the details of the Pull Request as per below.

Linus Hakansson / ExampleProjectSpace / ExampleProject / Pull requests

NewFeature

 newFeature → master **OPEN**


Created 39 seconds ago · Last updated 39 seconds ago

Edit Approve Merge ... ⚙️

- Updates and new Pipeline


> 0 attachments

0 comments



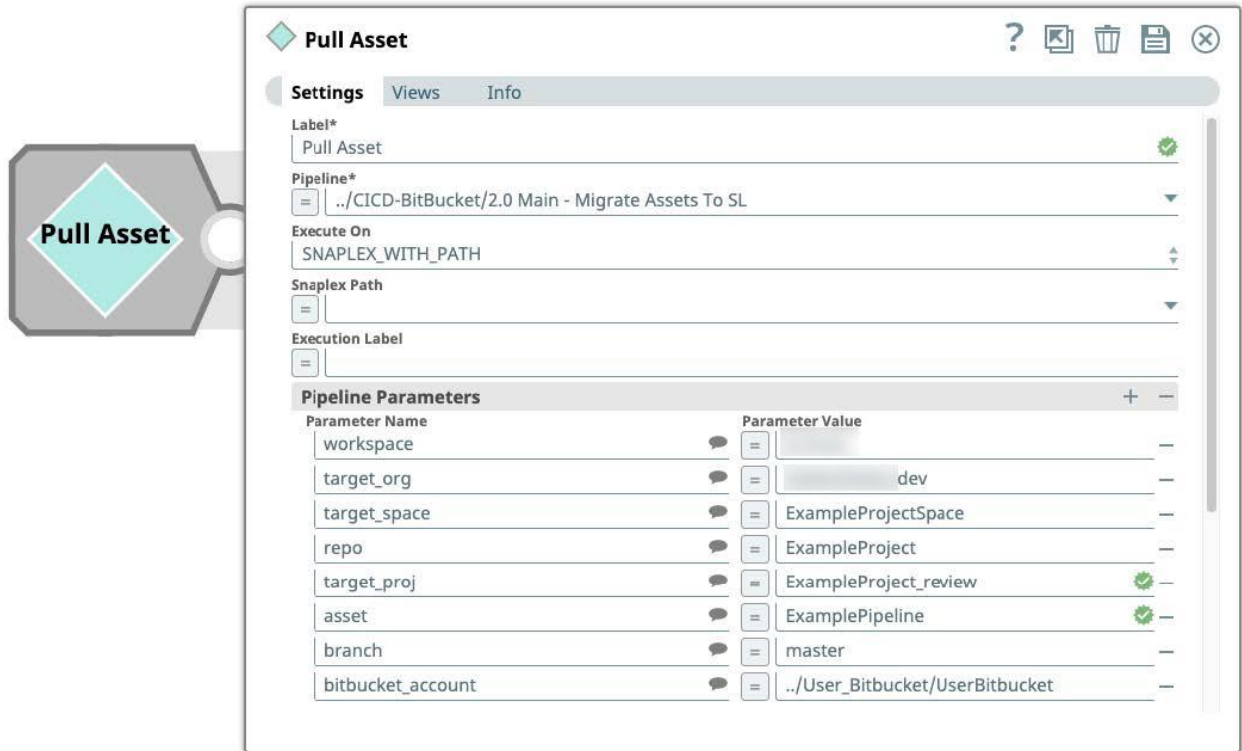
> 4 commits

3 files

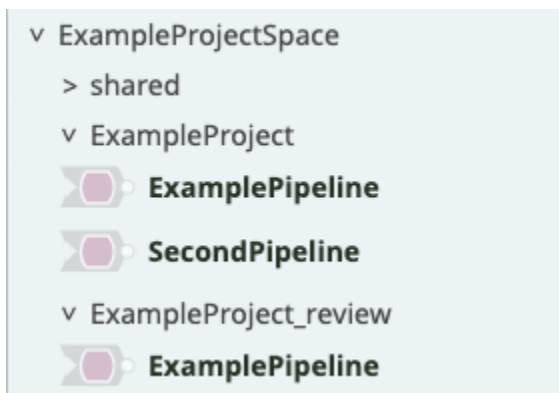
FILTER BY COMMENTS  SORT BY File tree ▾

Bitbucket allows configurations on Pull Request review so that the right notifications and processes can be embedded based on the organization requirements.

To review the changes of the Pull Request, we will focus on comparing the changes for the *ExamplePipeline* Pipeline. By using the Pull Asset Pipeline [described in a previous section](#), we can choose to copy the version of the Pipeline corresponding to the version on the master branch. Note that this can be changed to any other branch to use as the reference of the comparison. For *target_proj*, *ExampleProject_dev* has been configured to indicate that we want a new project created for the reference asset.



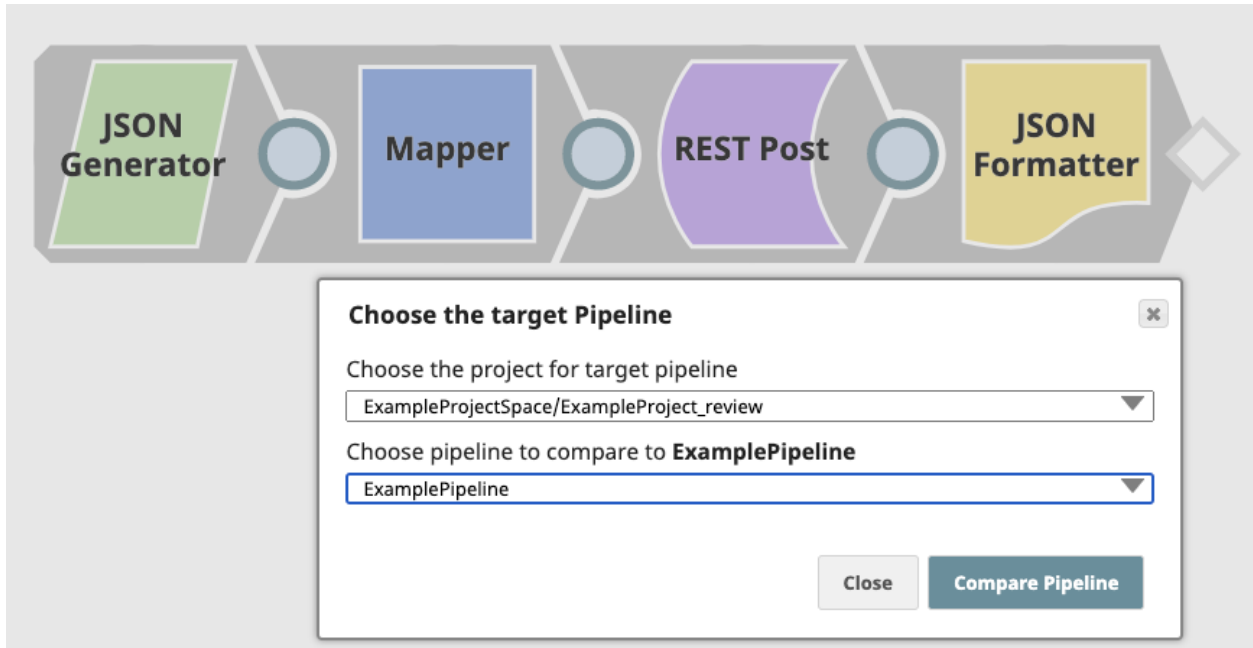
When the Pipeline has finished, we now have a new SnapLogic project that holds our single asset we want to compare to.



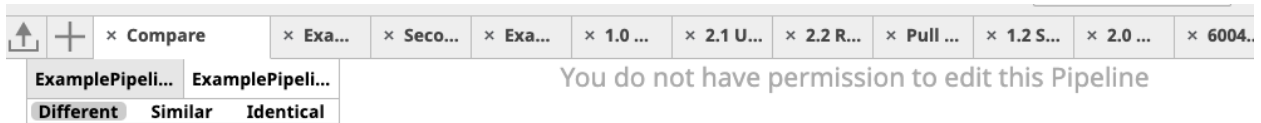
We can now open up our new Pipeline (that has the changes) and press the *Compare Pipeline* icon.



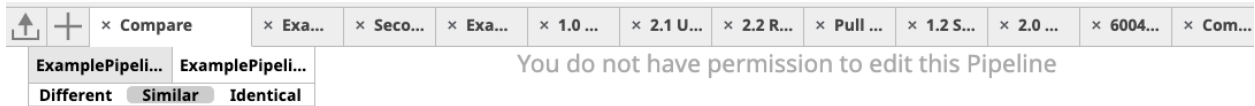
Here we navigate to the new ExampleProject_review project and select the ExamplePipeline (corresponding to master branch and CompanyGlobal_prod asset).



By selecting *Different*, we can see that we have added a new JSON Formatter Snap.



By selecting *Similar*, we can see that the Mapper has changed between the versions. Pressing on the Mapper Snap brings up a detailed property comparison between the two files.



Source: Mapper

Settings Views Info

Label
Mapper

Null-safe access

Pass through

Snap Execution

Validate & Execute

Transformations*

Mapping root
\$

Mapping table*

Expression*	Target path
\$msg	\$message
\$num	\$number

Target: Mapper

Settings Views Info

Label
Mapper

Null-safe access

Pass through

Snap Execution

Validate & Execute

Transformations*

Mapping root
\$

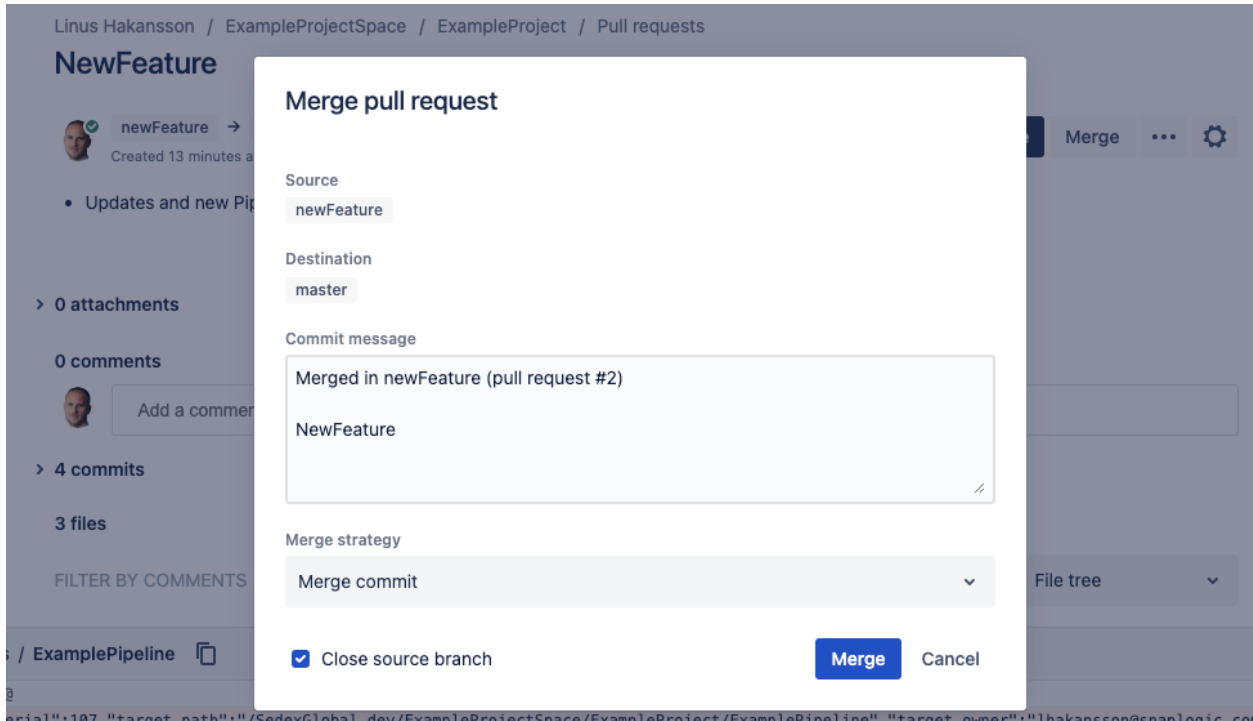
Mapping table*

Expression*	Target path
\$msg	\$message

Obviously, we could also choose to explore the SecondPipeline in detail, as well as potential changes in Accounts and Tasks, but that is out of the scope for this example.

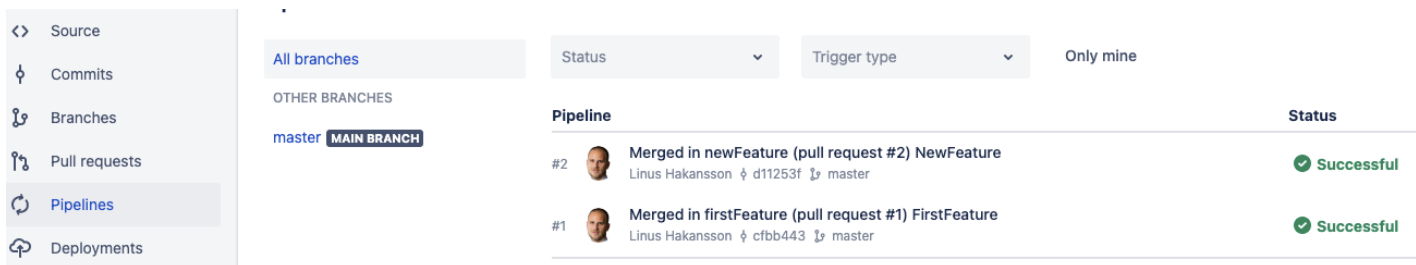
5.4 Approving and merging Pull Request

As our colleague has now reviewed the Pipeline, they will *Approve* and potentially (depending on process) *Merge* the Pull Request as per below.



In the above example, the *Close source branch* setting was enabled to have the generated branch deleted.

When the changes have been merged into the *master* branch, the Bitbucket Pipelines feature will pick up on the event thanks to [our CI/CD configuration](#). Navigating to the Pipelines section of the repository, we can see that a new Pipeline was just successfully executed.



This means that the PromotionRequest Pipelines was invoked by the Bitbucket Pipeline script through the Triggered Task, and the assets have now been created and updated in the production organization. This was described in details in a [previous section](#).

Navigating to the CompanyGlobal_prod organization, we can now see that our ExamplePipeline has been updated, while the SecondPipeline asset was created.

Snaps

Pipelines

Patterns



× ExamplePipe...

× LH new pipel...

× Compare

Enter Search Term

- > shared
- > API Management
- > Call To Action Integration
- ▼ ExampleProjectSpace
 - > shared
 - ▼ Example Project
 - ExamplePipeline
 - SecondPipeline

