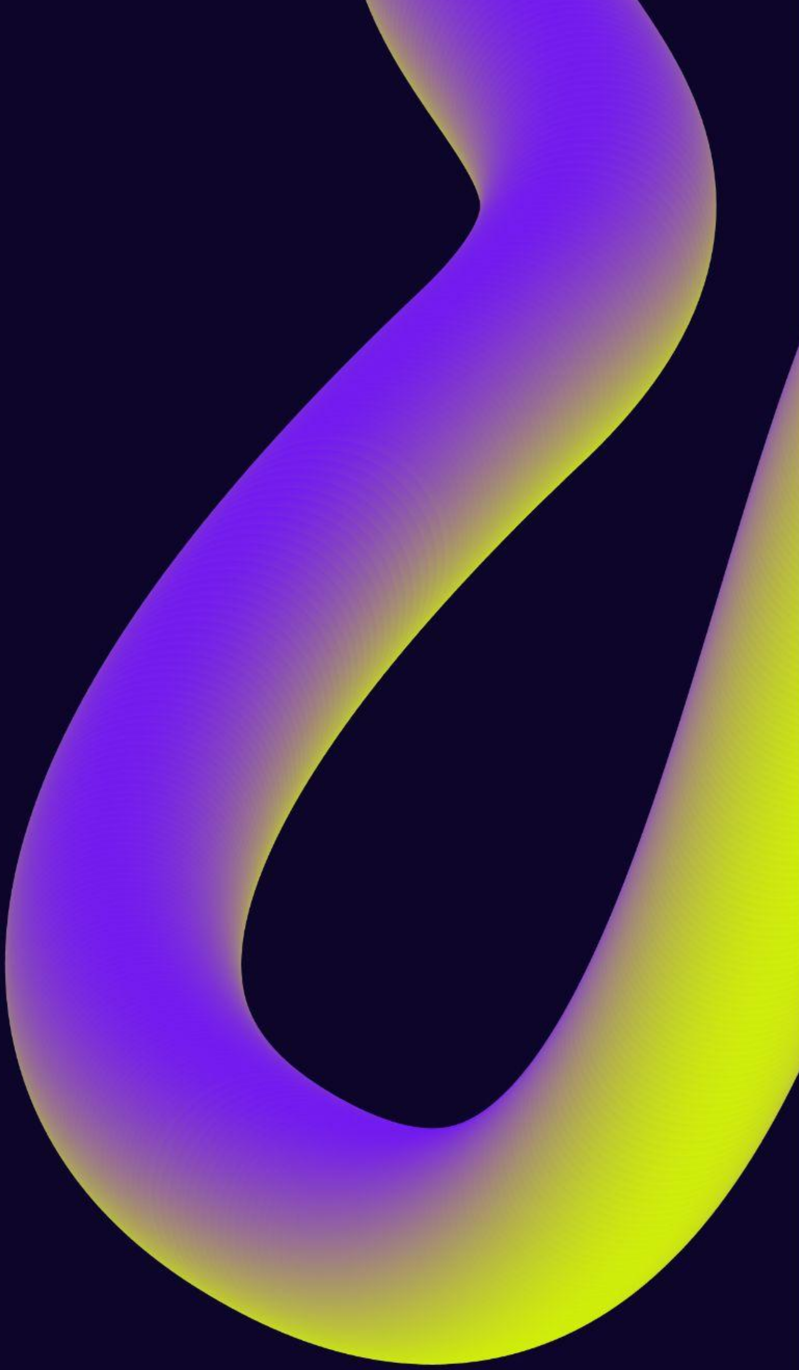
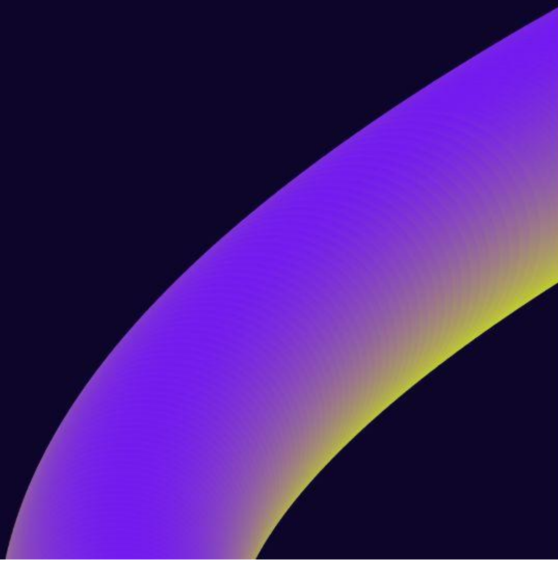
JELLYFISH

Agent Guide

December 2023





Agent Guide

Table of Contents

2 **Overview**

3 **Configuring the Agent**

- 3 Before you begin
- 11 System requirements
- 12 Configuring the Jellyfish Agent
- 34 Ensuring proper network configuration

35 **Running the Agent**

- 36 Docker run command syntax
- 37 Common run commands
- 40 Troubleshooting

41 **Reference**

- 41 Jira issue data
- 42 Config.yml examples
- 45 Agent usage modes

46 **Frequently Asked Questions**

Overview

Jellyfish is a SaaS application hosted in the AWS cloud. To provide a comprehensive view of your organization's engineering work, Jellyfish must integrate with Jira and your source control system. Jellyfish offers three integration options:

- Direct connection: Jellyfish Cloud connects to Cloud Customer Instances
- Direct connection: Jellyfish Cloud connects to On-Premise Customer Instances
- Indirect connection: Jellyfish Agent runs in Customer Environment, connects to Customer Instances, and uploads auditable data files to AWS Cloud

This Agent documentation will help you set up an indirect connection integration with one or more of your systems. It also includes instructions for running and maintaining the Agent.

To set up a direct connection, contact your Jellyfish representative.

How the Jellyfish Agent works

The Jellyfish Agent is an open source, on-premise tool that downloads Jira and git metadata from your cloud and on-premise systems and then packages and sends the data to the Jellyfish Cloud. The Jellyfish Agent runs inside your network environment and can be configured with fine-grained control over what data is sent to Jellyfish.

Configuring the Agent

Before you begin

To prepare for configuring the Jellyfish Agent, you need to:

- ➔ know a few things
- ➔ make some decisions
- ➔ gather some information

Things to know

This guide assumes:

- You're comfortable with a command line environment in whatever system you're using to run the Agent (most likely Linux), including the mechanics of running commands, setting environment variables, editing files, and so on.
- You're comfortable editing YAML files.

  Resources:

  https://yaml.org/

  https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started

Decisions to make

General decisions

➔ Where do you want to run the Jellyfish Agent?

The Agent can be installed on any server or virtual machine that meets system requirements.

➔ Which systems do you want to connect to Jellyfish via the Agent?

Jellyfish supports both on-premise and cloud options for the following systems:

- Jira
- GitHub
- Bitbucket
- Gitlab

➔ What data are you trying to send to Jellyfish? Which Jira projects and which git repos house the engineering activity you would like to track with Jellyfish?

Decisions around connecting Jira to the Agent

- What Jira projects do you want to track in Jellyfish, both now and in the future? Do you want future Jira projects to automatically flow to Jellyfish by default (and take action when you want to exclude a project) or, do you NOT want them to flow to Jellyfish by default and (take action when you want to include a new project)?
- Is there a subset of issues within the projects that are important to exclude from Jellyfish?
- Do you want to send only named issue data to Jellyfish (and take action when you want more), or do you want to send everything by default (and take action when you want to limit what is sent to Jellyfish)?
- If your Jira Users include customer records, do you want to filter the users tracked in Jellyfish by required email domain? And if yes, what about Jira Users without email addresses - do you want to include them too?

Decisions around connecting GitHub to the Agent

- Consider additional repos that may show up in the future. Do you want things to flow automatically to Jellyfish by default (and take action when you want to exclude a repo), or do you NOT want things to flow automatically (and take action when you want to include a repo)?
- Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs, you may want Jellyfish to crawl additional branches for commits. Jellyfish automatically pulls commits on the default branch of each repo (“main” or “master” is typically configured as the default branch). Do you need the Agent to crawl additional branches for commits?

Decisions around connecting Bitbucket to the Agent

- Where (projects, repos) is the work that you want to track in Jellyfish happening?
- Consider additional projects/repos that may show up in the future. Do you want things to flow automatically to Jellyfish by default (and take action when you want to exclude a project/repo) or, do you NOT want things to flow automatically (and take action when you want to include a new project/repo)?
- Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs, you may want Jellyfish to crawl additional branches for commits. Jellyfish automatically pulls commits on the default branch of each repo (“main” or “master” is typically configured as the default branch). Do you need the Agent to crawl additional branches for commits?

Decisions around connecting GitLab to the Agent

- Where (top-level groups, sub-groups, repos) is the work that you want to track in Jellyfish happening?
- Consider additional top-level groups that may show up in the future. Do you want things to flow automatically to Jellyfish by default (and take action when you want to exclude a top-level group) or, do you NOT want things to flow automatically (and take action when you want to include a new top-level group)?
- Similarly, do you want things to flow automatically from future sub-groups by default? Do you want things to flow automatically from future repos by default?
- Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs, you may want Jellyfish to crawl additional branches

for commits. Jellyfish automatically pulls commits on the default branch of each repo ("main" or "master" is typically configured as the default branch). Do you need the Agent to crawl additional branches for commits?

Information to gather

To set up the Jellyfish Agent, you must gather the following information:

- Your Jellyfish API token
- API URL, credentials/access token, and additional information for each system you're connecting to Jellyfish via the Agent.

Jellyfish API token

Ask your Jellyfish representative for an API token. Either they will send you the token securely or they will give you access to your organization's Jellyfish account. If you are given account access, locate the token on the Data Connections page:

1. Log into Jellyfish.
2. On the navigation toolbar, go to **Settings > System Setup > Data Connections**.

Jira information

- URL to your Jira Instance (e.g. https://jira.company.com, or https://company.atlassian.net)
- Username

  Note: The username must belong to a group that has the global **Browse Users** and **Groups** permission. Moreover, the account must have project-level access to view issues (**Browse Project** and **View Issue**) for all engineering-relevant projects.
- User password/token

If connecting this	Then gather this
Jira server with basic authentication disabled	A personal access token **Note**: To determine if basic authentication is disabled, refer to the “How to test if my basic authentication is disabled” section of the Atlassian article, Disabling basic authentication. **Note**: To create a personal access token, refer to the Atlassian article, Using Personal Access Tokens.
Jira server with basic authentication NOT disabled	The password associated with the above username
Jira cloud	A personal API token **Note**: To create a personal API token, refer to the Atlassian article, Manage API tokens for your Atlassian account.

- IF you’re sending only a subset of your Jira projects to Jellyfish, gather the Project Keys of those projects.
- Issue fields you’re NOT sending to Jellyfish if any.

  IMPORTANT! Jellyfish depends on certain issue fields in order to work properly. See Jira issue data. Excluding this required metadata can lead to broken functionality in the Jellyfish application if not carefully configured; please review any configuration of this type with your Jellyfish Representative.

Source control information

IMPORTANT! If you’re connecting multiple instances of source control, for example, Bitbucket and GitHub or two Bitbuckets, in addition to collecting the information in the table, you must obtain a **Git Instance Slug** for each instance from your Jellyfish representative.

System	Information you need to collect
GitHub server/on-premise	● GitHub URL (https://github.yourcompany.com/api/v3) ● Access token ● Organization Name ● If you want to pull from a fixed set of repos and NOT pull from other repos that may show up later, gather those repo names ● If you want to omit a fixed set of repos and automatically pull from repos that may show up later, gather those repo names **Note**: Repo names are not necessary if you want the Agent to pull from all repos.
GitHub cloud	● GitHub API Endpoint (https://api.github.com/) ● Access token ● Organization Name ● If you want to pull from a fixed set of repos and NOT pull from other repos that may show up later, gather those repo names ● If you want to omit a fixed set of repos and automatically pull from repos that may show up later, gather those repo names **Note**: Repo names are not necessary if you want the Agent to pull from all repos.

System	Information you need to collect
BitBucket server/on-premise	● BitBucket URL (https://bitbucket.yourcompany.com) ● Username ● Password ● Organization Name as it appears in the URL ● If you want to pull from a fixed set of projects and NOT pull from other projects that may show up later, gather the Project Keys of the projects you want to pull from. ● If you want to omit a fixed set of projects and automatically pull from projects that may show up later, gather the Project Keys of the ones you want to exclude. ● If you want to pull from a fixed set of repos or omit a fixed set of repos, gather the repo names. **Notes**: Project keys are not necessary if you want the Agent to pull from all projects. Repo names are not necessary if you want the Agent to pull from all repos.
BitBucket cloud	● Bitbucket API Endpoint (https://api.bitbucket.org) ● Username ● App password ● Organization Name as it appears in the bitbucket URL (bitbucket.org/organizationname) If your repos are grouped into projects: ● If you want to pull from a fixed set of projects and NOT pull from other projects that may show up later, gather the Project Keys of the projects you want to pull from. ● If you want to omit a fixed set of projects and automatically pull from projects that may show up later, gather the Project Keys of the ones you want to exclude. Similarly, ● If you want to pull from a fixed set of repos or omit a fixed set of repos, gather the repo names. **Notes**: Project keys are not necessary if you want the Agent to pull from all projects. Repo names are not necessary if you want the Agent to pull from all repos.

System	Information you need to collect
GitLab server/on-premise and cloud	● GitLab URL ● User access token (NOT a group access token) **IMPORTANT!** The user access token must have access to all repos that you want Jellyfish to evaluate. If multiple user groups dictate repo access, we recommend that you do the following: 1. Create a new Gitlab user account for Jellyfish. 2. Do one of the following: ● Add the new user to the groups with the repos you want Jellyfish to evaluate. ● Create a new group, add the new user to the group, and give the group access to each of the repos you want Jellifish to evaluate. ● Top-level Group IDs ● Sub-group IDs (not needed if pulling from all sub-groups) ○ If you want to pull from a fixed set of sub-groups and NOT pull from other sub-groups that may show up later, gather their Group IDs. ○ If you want to omit a fixed set of sub-groups and automatically pull from sub-groups that may show up later, gather their Group IDs. ● Project IDs (not needed if pulling from all repos) ○ If you want to pull from a fixed set of repos and NOT pull from other repos that may show up later, gather the Project IDs of the repos. ○ If you want to omit a fixed set of repos and automatically pull from repos that may show up later, gather the Project IDs of the repos.

System requirements

The Jellyfish Agent can be installed on any server or virtual machine that meets the following minimum requirements:

- 2 GB of RAM
- 20 GB of storage
- 2 processor cores
- Has network access to the systems you're connecting to
- Is running Docker

Note: The instructions in this document assume a relatively modern docker that supports `--mount` syntax. If a version that supports `--mount` syntax is not available, refer to this Docker document to learn how to use `-v` instead.

Configuring the Jellyfish Agent

The Jellyfish Agent extracts Jira and git metadata from your cloud and on-premise systems and packages and sends the data to the Jellyfish API.

In this procedure, you'll create and save two files:

- **Environment variables (cred.env) file**. This file stores the credentials the Agent will use to access the systems you're connecting to Jellyfish via the Agent.
- **Config.yml file**. This document contains your instructions for the Agent. It tells the Agent, among other things, how to access the systems you want to connect (aka, their URLs), what metadata you want to be pulled from the system, and from which projects/repositories/groups you want the data pulled.

In this section

Step 1: Store credentials in a cred.env file

Step 2: Copy the sample config.yml file

Step 3: Modify and save the config.yml file

What's next?

Step 1: Store credentials in a cred.env file

1. Create an environment variables file.
2. Add the Jellyfish API Token to the file in the following format:

   ```
   JELLYFISH_API_TOKEN=<token>
   ```

   Note: If you do not have the token, ask your Jellyfish representative. They will either send you the token securely or give you access to your organization's Jellyfish account. If you are given account access, your token can be found on the Data Connections page:

   1. Log into Jellyfish.
   2. On the navigation toolbar, go to **Settings > System Setup > Data Connections**.

3. Are you connecting Jira to Jellyfish via the Agent?
    - If no, skip this step.
    - If yes, add your Jira credentials to the file in the following format:

If you're connecting this	Then add these credentials
Jira server with basic authentication disabled	JIRA_USERNAME=<username> JIRA_BEARER_TOKEN=<BearerToken>
Jira server with basic authentication NOT disabled	JIRA_USERNAME=<username> JIRA_PASSWORD=<PasswordAssociatedWithUsername>
Jira Cloud (atlassian.net)	JIRA_USERNAME=<username> JIRA_PASSWORD=<PersonalAPIToken>

4. Are you connecting any source control systems to the Agent?
    - If no, skip to Step 6.
    - If yes, for each source control you're connecting to the Agent, add the applicable credential fields and values to the file.

System	Add this text
Bitbucket	BITBUCKET_USERNAME BITBUCKET_PASSWORD
GitHub	GITHUB_TOKEN=<token>
GitLab	GITLAB_TOKEN=<PersonalAccessToken>

5. Did you add credentials for multiple source control instances?
    - If yes, add an **ORG1_** prefix to the credentials associated with the first source control instance listed, add an **ORG2_** prefix to the credentials associated with the second source control instance listed, and so on, until every git credential has a prefix. See the examples below.
    - If no, skip this step.

Examples

Jellyfish, Jira, and one instance of GitLab

```
JELLYFISH_API_TOKEN=<token>
JIRA_USERNAME=<username>
JIRA_PASSWORD=<passwordassociatedwithusername>
GITLAB_TOKEN=<personalaccesstoken>
```

Jellyfish, Jira, and two Bitbucket instances (ORG1 and ORG2)

```
JELLYFISH_API_TOKEN=<token>
JIRA_USERNAME=<username>
JIRA_PASSWORD=<passwordassociatedwithusername>
ORG1_BITBUCKET_USERNAME=<username>
ORG1_BITBUCKET_PASSWORD=<password>
ORG2_BITBUCKET_USERNAME=<username>
ORG2_BITBUCKET_PASSWORD=<password>
```

Jellyfish, Jira, and one Bitbucket instance (ORG1) and one GitHub instance (ORG2)

```
JELLYFISH_API_TOKEN=<token>
JIRA_USERNAME=<username>
JIRA_PASSWORD=<passwordassociatedwithusername>
ORG1_BITBUCKET_USERNAME=<username>
ORG1_BITBUCKET_PASSWORD=<password>
ORG2_GITHUB_TOKEN=<token>
```

Adjusting the timeout duration before the agent quits and rolls back to stable

```
# OVERRIDE_TIME_LIMIT=<duration><unit (hour|minute|second)> eg ;
OVERRIDE_TIME_LIMIT=6h
```

Step 2: Copy the sample config.yml file

1. Access the sample config.yml file.
2. Copy the file.

Step 3: Modify the config.yml file

The sample config.yml file contains parameters that control various aspects of the Agent's behavior such as which systems/URLs to connect to, which projects/organizations/groups to pull data from, and even what data to pull. The file contains all the instructions the Agent needs to connect to and pull data from Jira and the source control systems Jellyfish supports.

Structure of the config.yml file

The sample config.yml file is divided into four sections.

- 1. Global configuration
- 2. Jira configuration
- 3a. Single instance Git configuration
- 3b. Multiple instance Git configuration

Resources for working with YAML files:

https://yaml.org/

https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started

How to modify the config.yml file

Because the Agent ignores all commented lines and follows the instructions of the uncommented lines, modifying the config.yml file entails the following:

1. Uncommenting ('commenting out') the sections of the file that do not apply to your configuration.
2. Updating the sections of the file that apply to the systems you want the Agent to connect to. This may entail commenting parameter lines, uncommenting parameter lines, updating parameter values, or any combination of these. Use the information in the tables in the sections that follow to guide your updates.

Guidelines

If you want to connect this to the Agent	Then comment out all lines in these sections	And update these sections of the file
Jira only	3a and 3b	1 and 2
Jira and a single source control instance	3b	1, 2, and 3a
A single git instance only (Jira connected directly)	2 and 3b	1 and 3a
Jira and multiple git instances	3a	1, 2, and 3b
Multiple git instances only (Jira connected directly)	2 and 3a	1 and 3b

Commenting out sections of the file

YAML does not support block comments. Therefore, **to comment out a section of the file, you must comment out EVERY line within the section.** To comment out a line, add a pound sign # and a space to the start of the line.

Note: Some lines within a section may already be commented out.

Updating section 1. Global configuration

1. Do any of the system servers your connecting to the Agent have a custom certificate?
    - If yes, do one of the following:
        - Make the custom certificate trustable and set the **`no_verify_ssl`** parameter (Line 10) to **`False`**.

          Note: To make the customer certificate trustable, refer to "Using the agent with custom SSL/TLS certificates" in the README file.
        - Set the **`no_verify_ssl`** parameter to **`True`**.
    - If no, set the **`no_verify_ssl`** parameter to **`False`**.
2. Do you want to send your config.yml file to Jellyfish?

   Note: Jellyfish uses the config.yml file as a diagnostic tool when assisting customers.
    - If yes, set the **`send_agent_config`** parameter to **`True`**.
    - If no, set the **`send_agent_config`** parameter to **`False`**.
3. Do you want to set a manual time limit for runs?

- If yes, set the **OVERRIDE_TIME_LIMIT** parameter to a time formatted as <amount><unit>

  Examples: **`30s`** for 30 seconds, **`24h`** for 24 hours, etc
- If no, the agent will use a time out value derived from the historical length of agent runs, retrieved from Jellyfish at the start of the run.

Updating section 2. Jira configuration

This section of the sample config.yml is section 2. Jira configuration.

Follow these steps if you are connecting Jira to Jellyfish via the Agent:

1. On Line 21, update the **`url`** value with your company's Jira URL.
2. Does your Jira instinct project data per GDPR?
   - If yes, set the **`gdpr_active`** parameter to **`True`**.
   - If no, set the **`gdpr_active`** parameter to **`False`**.

   Note: As of 9/2022, this should be **`True`** for Jira Cloud and **`False`** for Jira Server.
3. Set the value of the **`earliest_issue_dt`** parameter to three months prior to today's date.

   Recommendation: We recommend sending issues to Jellyfish in three-month batches initially. Set this value to three months ago, run the agent, then set the value to six months ago, run the agent, and so on, until Jellyfish has all the issues of the desired time frame. Because Jellyfish only ingests changes, only three months of issues are ingested with each Agent run.
4. Leave the default values for the **`issue_download_concurrent_threads`** and **`issue_batch_size`** parameters.

   Note: These parameters are included in the config.yml file to troubleshoot internal server errors. If when you run the Agent you get such an error, lower one or both of these values.
5. Do you want to send worklogs (self-reported hours logged against tickets) to Jellyfish? This can be a very large amount of data and does not normally show up in the app anyway. Consult a Jellfyfish Representative if you're thinking about enabling this. Otherwise, leave the download_worklogs parameter set to **`False`**.
6. Leave the **`download_sprints`** parameter set to **`True`**.

   Note: If you run the Agent and think that the Jira pull is taking too long, change this value to False and run the Agent again.

7. Do you want to pull data from all projects or only specific projects?
    - If all projects, comment out the **`include_projects`** parameter and value lines.
    - If specific projects, do **one** of the following:

If you want the Agent to pull data from these projects	Then follow these steps	Example

Specific projects only	Delete the **`include_projects`** values (PROJ1, PROJ2) and add the Project Keys of the projects you want the Agent to pull from, one per line.	`include_projects:`
		`  - FELIX`
		`  - AXEL`
		`  - AUTHX`

All projects with some exceptions	1. Comment out the **`include_projects`** parameter and value lines.
	2. Uncomment the **`exclude_projects`** parameter and value lines
	3. Delete the **`exclude_projects`** value (PROJ1) and add the Project Keys of the projects you don't want the Agent to pull from, one per line.
		`#  include_projects:`
		`#    - PROJ1`
		`#    - PROJ2`
		
		`  exclude_projects:`
		`    - BRIDGE`

All projects within certain categories only	1. Comment out the **`include_projects`** parameter and value lines.
	2. Uncomment the **`include_project_categories`** parameter and value lines.
	3. Add the project categories, one per line.
		`#  include_projects:`
		`#    - PROJ1`
		`#    - PROJ2`
		
		`include_project_categories:`
		`  - Engineering`

All projects within all but certain project categories	1. Comment out the **`include_projects`** parameter and value lines.
	2. Uncomment the **`exclude_project_categories`** parameter and value lines.
	3. Add the project categories, one per line.
		`#  include_projects:`
		`#    - PROJ1`
		`#    - PROJ2`
		
		`exclude_project_categories:`
		`  - Support`

If you want the Agent to pull data from these projects	Then follow these steps	Example

All projects within certain categories except certain projects	1. Comment out the **`include_projects`** parameter and value lines.
	2. Uncomment the **`include_project_categories`** parameter and value lines.
	3. Add the project categories, one per line.
	4. Uncomment the **`exclude_projects`** parameter and value lines.
	5. Delete the value (PROJ1) and add the Project Keys of the projects within the included categories that you don't want the Agent to pull from.
		`#  include_projects:`
		`#    - PROJ1`
		`#    - PROJ2`
		
		`exclude_projects:`
		`    - BRIDGE`
		
		`include_project_categories:`
		`  - Engineering`

IMPORTANT! The Agent only pulls data from projects that meet ALL criteria. For instance, if you enter values for **`include_projects`** and **`include_project_categorie`**s, the Agent will only pull from the subset of named projects in the specified categories. Include_projects and include_project_categories **cannot** be used to include all projects in named categories plus additional projects outside the named categories.

8. Do you want to pull all issues from the projects or only those that match a specific Jira Query Language(JQL) filter?
   - If all issues, skip this step.
   - If only those that match a specific JQL filter, uncomment the **`issue_jql`** parameter and enter the JQL query.
9. Do you want the Agent to pull all issue fields or only specific fields?

   WARNING: Certain issue fields are required for Jellyfish to work. For a list of these fields, refer to Jira issue data.
   - If no, skip this step.
   - If yes, do one of the following:

If you want the Agent to pull	Then follow these steps	Example

All issue fields except certain ones	1. Uncomment the **`exclude_fields`** parameter and value lines.
	2. Delete the **`exclude_fields`** values and add the issue fields you want to be excluded, one per line.
	
	⚠ **WARNING** Do NOT exclude any fields required by Jellyfish; you may only exclude optional fields. Required fields are listed in the Jira issue data topic.
		`exclude_fields:`
		`  - summary`
		`  - description`
		`  - worklog`
		`  - comment`

Certain issue fields only	1. Uncomment the **`include_fields`** parameter and value lines.
	2. Delete the include_fields values and add the issue fields, one per line.
	
	⚠ **WARNING** At a minimum, you MUST include all fields required by Jellyfish. These are listed in the Jira issue data topic.
	
	The example includes all required fields plus two optional fields, *labels* and *duedate*.
		`include_fields:`
		`  - id`
		`  - subtasks`
		`  - resolutiondate`
		`  - issuekey`
		`  - issuetype`
		`  - created`
		`  - updated`
		`  - project`
		`  - issuelinks`
		`  - parent`
		`  - status`
		`  - assignee`
		`  - creator`
		`  - reporter`
		`  - resolution`
		`  - customfield_10001 # Epic Link`
		`  - customfield_10002 # Epic Name`
		`  - customfield_10003 # Sprint`
		`  - customfield_10004 # Parent Link`
		`  - customfield_10005 # Story Points`
		`  - customfield_10008 # Rank`
		`  - labels`
		`  - duedate`

10. If your Jira Users include customer records, do you want to filter users sent to Jellyfish by a specific set of required email domains?
    - If no, then section 2 of the config.yml file is complete.
    - If yes, do the following:
        1. Uncomment the **required_email_domains** parameter and value lines.
        2. Replace jellyfish.co with the first required email domain.
        3. Add any additional required email domains to the list.
        4. Do you want to send the Jira Users that don't have an email address to Jellyfish?
            - If yes, then section 2 of the config.yml file is complete.
            - If no, uncomment **is_email_required = True**.

Updating section 3a. Single instance configuration

Follow the instructions in the section that applies to the git source control you're connecting to the Agent.

- Single GitHub instance
- Single Bitbucket Cloud instance
- Single Bitbucket Server instance
- Single GitLab instance

Single GitHub instance

1. Set the **`provider`** parameter to **`github`**.
2. Update the value of the **`url`** parameter.
    - For GitHub Cloud, set the value to **`https://api.github.org`**.
    - For GitHub Enterprise, set the value to **`https://github.YOURCOMPANY.com/api/v3`**, where YOURCOMANY is the name of your company.
3. To the **`include_projects`** parameter, replace **`PROJ1`** with the Organization name as it appears in the GitHub URL.
4. Do one of the following:

If you want the Agent to pull from	**Then do this**	**Example**

All repositories	Comment out the **`include_repos`** parameter and value lines.	`# include_repos:`
		`#   - my_repository`

Certain repositories only	Delete the **`include_repos`** value (my_repository) and add the names of the repositories you want the Agent to pull from, one per line	`include_repos:`
		`  - client_config`
		`  - product`
		`  - automation`

If you want the Agent to pull from	Then do this	Example

All but certain repositories	1. Comment out the **`include_repos`** parameter and value lines.
	2. Uncomment the **`exclude_repositories`** parameter and value lines.
	3. Delete the value (repo_to_skip) and add the names of the repositories you want to be excluded, one per line.
		`# include_repos:`
		`#   - my_repository`
		
		`exclude_repos:`
		`  - services`

5. Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs and want Jellyfish to crawl additional branches for commits, you can specify these in the config.yml file. Jellyfish automatically pulls commits on the default branch of each repo ("main" or "master" is typically configured as the default branch). Do you want the Agent to pull from any other branches?
    a. If no, skip this step.
    b. If yes, do the following:

Steps	Example

1. Uncomment the **`include_branches`**, **`my_repository`**, **`- master`**, and **`- develop`** lines.
2. Replace **`my_repository`** with the name of the repository you want to select an additional branch for.
3. Replace **`master`** and **`develop`** with the names of your default branches and additional branches.
4. Add additional repository names and branches as needed.
	`include_branches:`
	`  client_config:`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`
	`  product`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`
	`        - alpha`
	`  automation`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`

6. Do you want the Agent to remove long-form text content (commit messages, PR text, etc.) from the data?
    a. If yes, update the **`strip_text_content`** parameter's value to **`True`**.
    b. If no, skip this step.
7. Do you want the Agent to redact the names and URLs of projects, repos, and branches?
    a. If yes, update the **`redact_names_and_urls`** parameter's value to **`True`**.
    b. If no, skip this step.
8. Comment out the **`verbose`** parameter.

Single Bitbucket Cloud instance

1. Set the **`provider`** parameter to **`bitbucket_cloud`**.
2. Set the **`url`** parameter to **`https://api.bitbucket.org`**.
3. To the **`include_projects`** parameter, replace **`PROJ1`** with the Organization name as it appears in your Bitbucket Cloud URLs.
4. Are your repos grouped into Bitbucket Cloud projects?
    - If no, skip this step.
    - If yes, do one of the following:

If you want the Agent to pull from these projects	**Then do this**	**Example**

All projects	Uncomment the **`include_bitbucket_cloud_projects`** parameter line ONLY; do not uncomment the value.	`include_bitbucket_cloud_projects:`
		`#   - PROJ1`

Certain projects only	1. Uncomment the **`include_bitbucket_cloud_projects`** parameter and value lines.
	2. Add the Project Keys of the projects you want the Agent to pull from, one per line.
		`include_bitbucket_cloud_projects:`
		`    - 123`
		`    - 456`
		`    - 789`

If you want the Agent to pull from these projects	Then do this	Example

All but certain projects	1. Uncomment the **`exclude_bitbucket_cloud_projects`** parameter and value lines.
	2. Add the Project Keys of the projects you don't want the Agent to pull from, one per line.
		`exclude_bitbucket_cloud_projects:`
		`    - 124`
		`    - 496`

5. Do you want the Agent to pull from all repositories or only specific ones?

If you want the Agent to pull from	Then do this	Example

All repositories	Comment out the **`include_repos`** parameter and value lines.	`# include_repos:`
		`#   - my_repository`

Certain repositories only	Delete the **`include_repos`** value (my_repository) and add the names of the repositories you want the Agent to pull from, one per line	`include_repos:`
		`  - client_config`
		`  - product`
		`  - automation`

All but certain repositories	1. Comment out the **`include_repos`** parameter and value lines.
	2. Uncomment the **`exclude_repositories`** parameter and value lines.
	3. Delete the value (repo_to_skip) and add the names of the repositories you want to be excluded, one per line.
		`# include_repos:`
		`#   - my_repository`
		
		`exclude_repos:`
		`  - services`

6. Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs and want Jellyfish to crawl additional branches for commits, you can specify these in the config.yml file. Jellyfish automatically pulls commits on the default branch of each repo ("main" or "master" is typically configured as the default branch). Do you want the Agent to pull from any other branches?
   - If no, skip this step.
   - If yes, do the following:

Steps	**Example**

1. Uncomment the **`include_branches`**, **`my_repository`**, **`- master`**, and **`- develop`** lines.
2. Replace **`my_repository`** with the name of the repository for which you want to select an additional branch.
3. Replace **`master`** and **`develop`** with the names of your default branches and additional branches.
4. Add additional repository names and branches as needed.
	`include_branches:`
	  `client_config:`
	      `- master`
	      `- main`
	      `- dev`
	      `- develop`
	  `product`
	      `- master`
	      `- main`
	      `- dev`
	      `- develop`
	      `- alpha`
	  `automation`
	      `- master`
	      `- main`
	      `- dev`
	      `- develop`

7. Do you want the Agent to remove long-form text content (commit messages, PR text, etc.) from the data?
   - If yes, update the **`strip_text_content`** parameter's value to **`True`**.
   - If no, skip this step.
8. Do you want the Agent to redact the names and URLs of projects, repos, and branches?
   - If yes, update the **`redact_names_and_urls`** parameter's value to **`True`**.
   - If no, skip this step.
9. Do you want more verbose local logging?
   - If yes, update the value of the **`verbose`** parameter to **`True`**.
   - If no, skip this step.

Single Bitbucket Server instance

1. Update the **`url`** parameter.
2. Do you want the Agent to pull from all projects or only specific ones?

If you want the Agent to pull from these projects	Then do this	Example

All projects	Comment out the **`include_projects`** parameter and values	`#  include_projects:`
		`#    - PROJ1`

Certain projects only	Delete the **`include_projects`** value (PROJ1) and add the Project Keys of the projects you want the Agent to pull from, one per line	`include_projects:`
		`  - 123`
		`  - 456`
		`  - 789`

All but certain projects	1. Comment out the **`include_projects`** parameter and values
	2. Uncomment the **`exclude_projects`** parameter and values
	3. Delete the **`exclude_projects`** value (PROJ1) and add the Project Keys of the projects you do NOT want the Agent to pull from, one per line
		`#  include_projects:`
		`#    - PROJ1`
		
		`exclude_projects:`
		`  - 321`
		`  - 987`

3. Do you want the Agent to pull from all repositories or only specific ones?

If you want the Agent to pull from	Then do this	Example

All repositories	Comment out the **`include_repos`** parameter and value lines.	`# include_repos:`
		`#   - my_repository`

Certain repositories only	Delete the **`include_repos`** value (my_repository) and add the names of the repositories you want the Agent to pull from, one per line	`include_repos:`
		`  - client_config`
		`  - product`
		`  - automation`

If you want the Agent to pull from	Then do this	Example

All but certain repositories	1. Comment out the **`include_repos`** parameter and value lines.
	2. Uncomment the **`exclude_repositories`** parameter and value lines.
	3. Delete the value (repo_to_skip) and add the names of the repositories you want to be excluded, one per line.
		`# include_repos:`
		`#   - my_repository`
		
		`exclude_repos:`
		`  - services`

4. Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs and want Jellyfish to crawl additional branches for commits, you can specify these in the config.yml file. Jellyfish automatically pulls commits on the default branch of each repo (“main” or “master” is typically configured as the default branch). Do you want the Agent to pull from any other branches?
    - If no, skip this step.
    - If yes, do the following:

Steps	Example

1. Uncomment the **`include_branches`**, **`my_repository`**, **`- master`**, and **`- develop`** lines.
2. Replace **`my_repository`** with the name of the repository for which you want to select an additional branch.
3. Replace **`master`** and **`develop`** with the names of your default branches and additional branches.
4. Add additional repository names and branches as needed.
	`include_branches:`
	`  client_config:`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`
	`  product`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`
	`        - alpha`
	`  automation`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`

5. Do you want the Agent to remove long-form text content (commit messages, PR text, etc.)from the data?
    - If yes, update the **`strip_text_content`** parameter's value to **`True`**.
    - If no, skip this step.
6. Do you want the Agent to redact the names and URLs of projects, repos, and branches?
    - If yes, update the **`redact_names_and_urls`** parameter's value to **`True`**.
    - If no, skip this step.
7. Do you want more verbose local logging?
    - If yes, update the value of the **`verbose`** parameter to **`True`**.
    - If no, skip this step.

Single GitLab instance

1. Set the **`provider`** parameter to **`gitlab`**.
2. Set the **`url`** parameter to one of the following:
    - Gitlab cloud: **`https://gitlab.com`**
    - Gitlab on-premise: **`https://gitlab.yourcompany.com`**
3. Replace the include_projects value (PROJ1) with the Group IDs of the top-level groups you want to pull from.

    Example

    ```
    include_projects:
      - 123
      - 456
      - 789
      - 626
    ```

    Note: If the number of groups you want to pull from is greater than the number of projects you don't want to pull from, consider using the **`exclude_projects`** parameter instead.

4. Do you want the Agent to pull from all the subgroups of the included top-level groups or only from specific ones?

If you want the Agent to pull from these subgroups	Then do this	Example

All subgroups	Uncomment the **`include_all_repos_inside_projects`** parameter ONLY	`include_all_repos_inside_projects:`
		`#    - PROJ1`

Certain subgroups only	1. Uncomment the **`include_all_repos_inside_projects`** parameter and values
	2. Delete the **`include_all_repos_inside_projects`** value (123) and add the Group IDs of the sub-groups you want the Agent to pull from, one per line
		`include_all_repos_inside_projects:`
		`  - 456`
		`  - 789`
		`  - 321`

All but certain subgroups	1. Comment out the **`include_all_repos_inside_projects`** parameter and values.
	2. Uncomment the **`exclude_all_repos_inside_projects`** parameter and values.
	3. Delete the **`exclude_all_repos_inside_projects`** value (123) and add the Group IDs of the subgroups you do NOT want the Agent to pull from, one per line.
		`#  include_projects:`
		`#    - PROJ1`
		
		`exclude_projects:`
		`  - 654`
		`  - 987`

5. Do you want the Agent to pull from all repositories or only specific ones?

If you want the Agent to pull from	Then do this	Example

All repositories	Comment out the **`include_repos`** parameter and value lines.	`# include_repos:`
		`#   - my_repository`

Certain repositories only	Delete the **`include_repos`** value (my_repository) and the **Project IDs** of the repos you want the Agent to pull from, one per line.	`include_repos:`
		`  - 956`
		`  - 385`
		`  - 823`
		`  - 145`

All but certain repositories	1. Comment out the **`include_repos`** parameter and value lines.
	2. Uncomment the **`exclude_repos`** parameter and value lines.
	3. Delete the value (repo_to_skip) and add the **Project IDs** of the repos you don't want the Agent to pull from, one per line.
		`# include_repos:`
		`#   - my_repository`
		
		`exclude_repos:`
		`  - 173`
		`  - 864`
		`  - 198`
		`  - 725`

6. Jellyfish pulls all activity associated with PRs. It can also crawl non-PR commits on particular named branches. If you have workflows that don't use PRs and want Jellyfish to crawl additional branches for commits, you can specify these in the config.yml file. Jellyfish automatically pulls commits on the default branch of each repo ("main" or "master" is typically configured as the default branch). Do you want the Agent to pull from any other branches?
   - If no, skip this step.
   - If yes, do the following:

Steps	Example

1. Uncomment the **`include_branches`**, **`my_repository`**, **`- master`**, and **`- develop`** lines.
2. Replace **`my_repository`** with the name of the repository for which you want to select an additional branch.
3. Replace **`master`** and **`develop`** with the names of your default branches and additional branches..
4. Add additional repository names and branches as needed.
	`include_branches:`
	`  client_config:`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`
	`  product`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`
	`        - alpha`
	`  automation`
	`        - master`
	`        - main`
	`        - dev`
	`        - develop`

7. Do you want the Agent to remove long-form text content (commit messages, PR text, etc.)from the data?
    - If yes, update the **`strip_text_content`** parameter's value to **`True`**.
    - If no, skip this step.
8. Do you want the Agent to redact the names and URLs of projects, repos, and branches?
    - If yes, update the **`redact_names_and_urls`** parameter's value to **`True`**.
    - If no, skip this step.
9. Comment out the **`verbose`** parameter.

Updating section 3b. Multiple instance configuration

This section of the config.yml file configures the "git" object as a list of objects.

For each source control instance you add to the list, do the following:

1. Add the following parameters:
   - `Provider`. Value options:
     - bitbucket_server
     - bitbucket_cloud
     - gitlab
     - github
   - `Url.` Value options:

Source control system	url
Github on-premise	https://github.YOURCOMPANY.com/api/v3 **Note**: There is no backlash at the end of this url.
Github cloud	https://api.github.org
Bitbucket on-premise	https://bitbucket.YOURCOMPANY.com
Bitbucket cloud	https://api.bitbucket.org
Gitlab on-premise	https://gitlab.YOURCOMPANY.com
Gitlab cloud	https://gitlab.com

   - `instance_slug`. Contact your Jellyfish representative if you have not already received this.
   - `creds_envvar_prefix`. In the cred.env file you created in Step 1, each set of git credentials has a unique prefix. Add the applicable prefix here.

     IMPORTANT! Make sure that the ORG prefix you enter is associated with the correct source control provider.

2. Follow the instructions in the appropriate section of Updating section 3a. Single instance to add additional parameters and values.

Ensuring proper network configuration

The Agent must be able to make requests to each of the systems it's connecting to. To do this, your network firewall/proxies must all these requests to get through. Therefore, as a final step in the configuration of the agent, verify that the agent is able to do the following:

- Make requests to the Jellyfish API at https://app.jellyfish.co:443/
- Make requests to your Jira and Git host(s) on port 443
- Make requests to URLs under s3.amazonaws.com on port 443

Running the Agent

Important: The instructions in this document assume a relatively modern docker that supports **`--mount`** syntax. If a version that supports **`--mount`** syntax is not available, refer to this Docker document to learn how to use **`-v`** instead.

Overview of the process

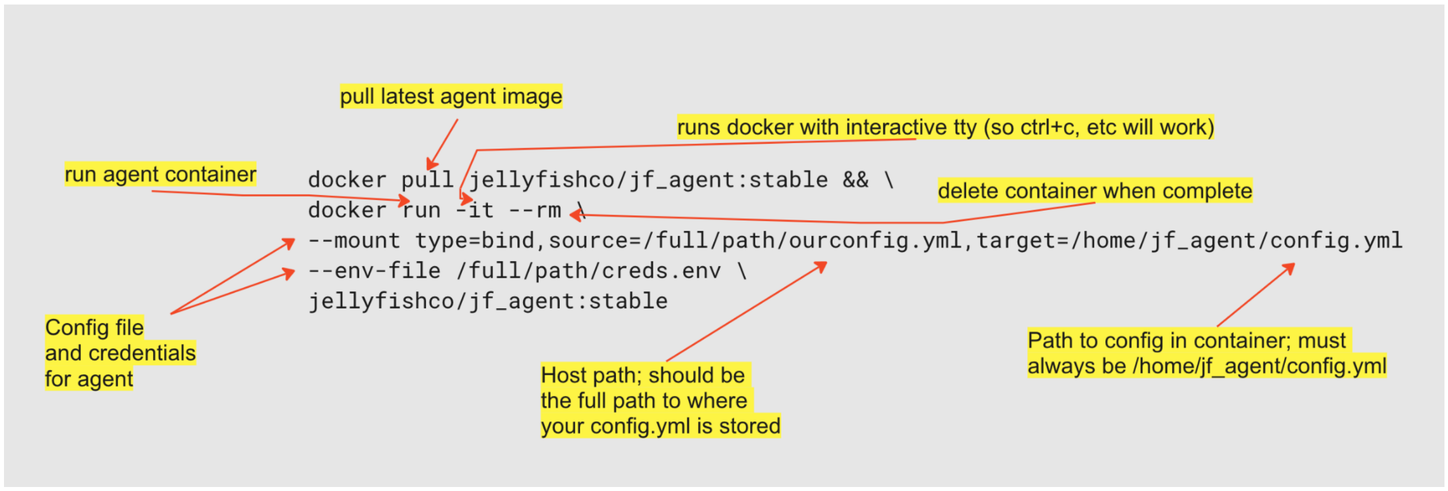
The Jellyfish Agent is distributed as a docker image. The image bundles the Agent's source code, a Python 3 environment, and the AWS command line tools. You'll execute a docker run command that includes your config.yml file and credentials. The docker run command creates a container for the config.yml file and credentials over the image on Docker Hub and then executes the command.

Automatic Rollback

If the agent exits with an error or hits a timeout, it will rollback to the release tagged `stable` in GitHub (irrespective of tagging in Dockerhub). This way, if a breaking change pushed in the latest release causes an unhandled exception or indefinite hang, the agent will still report data back for that day. The time limit for an agent run is derived from historical lengths of agent runs and retrieved from Jellyfish at the start of the run.

- The time limit can be overridden in config using `OVERRIDE_TIME_LIMIT`.
- Automatic rollback can be disabled using ROLLBACK_OVERRIDE (add to run command with –env ROLLBACK_OVERRIDE).

Docker run command syntax


pull latest agent image
runs docker with interactive tty (so ctrl+c, etc will work)
run agent container
delete container when complete
Config file and credentials for agent
Host path; should be the full path to where your config.yml is stored
Path to config in container; must always be /home/jf_agent/config.yml

```
docker pull jellyfishco/jf_agent:stable && \
docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml
--env-file /full/path/creds.env \
jellyfishco/jf_agent:stable
```

Using a bind mount to provide your config.yml file

The Agent is hard-coded to read from the path **`/home/jf_agent/config.yml`** in its container. This means that to get your config.yml file into the Agent, you need to mount your local config file in such a way that it appears under the hard-coded path inside the container.

To mount your config.yml file for reading:

```
--mount
type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml
```

Where **`/full/path/ourconfig.yml`** is a placeholder for the full path to your actual config.yml file.

Using a bind mount to provide a directory for output

The Agent is hard coded to write to the path **`/home/jf_agent/output`**. This means that if you want to see the output after it's written, you need to mount a directory from your host in such a way that the written files appear inside your directory. If you don't do this, then the output will not be accessible after the container exists.

To mount a directory for output:

```
--mount type=bind,source=/full/path/jf_agent_output,target=/home/jf_agent/output
```

Where **`/full/path/jf_agent_output`** is a placeholder for the full path to an existing directory you can write to where you want the output to go.

Important: The directory must already exist and be writeable.

Common run commands

download_and_send (normal mode)

This run command downloads Jira and/or git data from your systems and sends it to Jellyfish. Upon completion, the downloaded data is stored inside the container.

```
docker pull jellyfishco/jf_agent:stable && \
docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--env-file /full/path/creds.env \
jellyfishco/jf_agent:stable
```

Automate nightly runs of the Agent

Set up a **cron job** to run the Agent in download and send mode in nightly intervals.

download_only mode

Use this run command to download Jira and/or git data from your systems only. Unlike normal mode, the downloaded data is NOT sent to Jellyfish.

```
docker pull jellyfishco/jf_agent:stable && docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--mount type=bind,source=/full/path/jf_agent_output,target=/home/jf_agent/output \
--env-file ./creds.env \
jellyfishco/jf_agent:stable -m download_only
```

The `--rm` argument cleans up the container and data when the Agent is complete. If you want to save the downloaded output instead, add a bind mount that maps a host directory to the container's agent output directory as in the third line above.

send_only mode

Use this run command if you've run the agent in download_only mode and are ready to send the data to Jellyfish.

```
docker pull jellyfishco/jf_agent:stable && docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--mount type=bind,source=/full/path/jf_agent_output,target=/home/jf_agent/output \
--env-file ./creds.env \
jellyfishco/jf_agent:stable -m send_only -od ./output/20190822_133513
```

Provide a bind mount for the output directory (line 3 above) and the -od argument (line 5) to specify a path relative to the container's output directory that contains the data previously downloaded.

When the agent runs, it saves its downloaded data in a timestamped directory inside of /home/jf_agent/output. It shows the directory that the downloaded data is written to with a line like this:

```
Will write output files into ./output/20190822_133513
```

If, for example, an earlier run with download_only may have written its output file into ./output/20190822_133513 and the host directory /tmp/jf_agent/output had been mounted at /home/jf_agent/output, you'd use these arguments to send that data to Jellyfish:

```
--mount type=bind,source=/tmp/jf_agent_output,target=/home/jf_agent/output -m send_only -od
./output/20190822_133513
```

print_all_jira_fields mode

Use this run command to display the keys and fields names of all your custom Jira fields.

```
docker pull jellyfishco/jf_agent:stable && docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--env-file ./creds.env \
jellyfishco/jf_agent:stable -m print_all_jira_fields
```

print_apparently_missing_git_repos mode

Use this command to display the names and URLs of git repositories that may be missing from Jellyfish.

```
docker pull jellyfishco/jf_agent:stable && docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--env-file ./creds.env \
jellyfishco/jf_agent:stable -m print_apparently_missing_git_repos
```

validate mode

Use this run command to validate your configuration file.

```
docker pull jellyfishco/jf_agent:stable && docker run -it --rm \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--env-file ./creds.env \
jellyfishco/jf_agent:stable -m validate
```

bypass rollback on failure

Use this if you want to prevent the agent from rolling back to a known functioning version after a failure or timeout with `ROLLBACK_OVERRIDE` in `creds.env` or in command line incantation.

```
docker run --rm -it \
--mount type=bind,source=/full/path/ourconfig.yml,target=/home/jf_agent/config.yml \
--env-file ./creds.env --env ROLLBACK_OVERRIDE \
jellyfishco/jf_agent:stable
```

set a manual timeout for agent runs

The agent will automatically get a time limit from Jellyfish based on how long the previous runs took, but if you wish to specify your own time limit you can in the `creds.env` file. The format is <integer duration><unit>. Examples: `7h` for seven hours, `80m` for 80 minutes, etc.

```
OVERRIDE_TIME_LIMIT=3h
```

Troubleshooting

If you get this error:

```
unknown flag: --mount
See 'docker run --help'.
```

It means you're on an outdated docker. Use -v instead. To learn how to use -v, refer to this Docker document.

Reference

Jira issue data

Jellyfish's data models are built on Jira issue metadata using both out-of-the-box and custom fields. The Agent will send data from all issue fields unless you narrow the scope using the **`include_fields`** or **`exclude_fields`** parameters in the Jira configuration section of the config.yml file. Keep in mind that you can only narrow the scope of optional data; you cannot add required data to either of these parameters.

In this topic, we've also listed additional Jira data that, when provided, is displayed on the Issue page. If there are additional fields outside of these two lists that you want Jellyfish to display, contact your Jellyfish representative.

Data required by Jellyfish

- issuekey
- parent
- issuelinks
- project
- reporter
- assignee
- creator
- issuetype
- resolution
- resolutiondate
- status
- created
- updated
- subtasks
- Epic Link[cf]
- Epic Name[cf]
- Sprint[cf]
- Parent Link[cf]
- Story Points[cf]
- Rank[cf]

[cf] custom field. Custom fields have keys in the form, customfield_xxxxx, where the digits represented by xxxxx are different in each Jira installation. You can find the keys for your custom fields by running the agent in **`print_all_jira_fields`** mode.

IMPORTANT! Do NOT add required fields to the exclude_fields parameters in your config.yml file.

Additional data used by Jellyfish

If there is data for these fields, it will show up on the issue card.

- summary
- description
- priority
- worklog
- comment
- timetracking
- duedate
- labels
- fixVersions
- versions
- components
- timeestimate
- timeoriginalestimate
- timespent
- aggregatetimespent
- aggregatetimeoriginalestimate
- aggregatetimeestimate

Config.yml examples

[Jira and Bitbucket Cloud](#)

[Jira, Github, and GitLab](#)

Jira and Bitbucket Cloud

This customer's repos are grouped into Bitbucket Cloud projects.

Highlights

The Agent pulls:

- Required issue data (only) from all issues in three Jira projects
- Work data from the default branches of all repos in all but three Bitbucket cloud projects

```
global:
  no_verify_ssl: False
  send_agent_config: True
jira:
  url: https://baldwin.atlassian.net
  gdpr_active: True
  earliest_issue_dt: 2022-01-01
  issue_download_concurrent_threads: 10
  issue_batch_size: 100
  download_worklogs: True
  download_sprints: True
  include_projects:
    - BB
    - BCR
    - FGM
 include_fields:
    - id
    - subtasks
    - resolutiondate
    - issuekey
    - issuetype
    - created
    - updated
    - project
    - issuelinks
    - parent
    - status
    - assignee
    - creator
    - reporter
    - resolution
    - customfield_10001 # Epic Link
    - customfield_10002 # Epic Name
    - customfield_10003 # Sprint
    - customfield_10004 # Parent Link
    - customfield_10005 # Story Points
    - customfield_10008 # Rank
git:
  provider: bitbucket_cloud
  url: https://api.bitbucket.org
  include_projects:
      - baldwin
  exclude_bitbucket_cloud_projects:
      - ATL
      - FMX
  verbose: True
  strip_text_content: False
  redact_names_and_urls: False
```

Jira, Github, and GitLab

Highlights

The Agent pulls:

- All issue data from all issues in all Jira projects
- Work data from the default branches of four specific repos in GitHub
- Work data from the default branches of three repos (project IDs) in three specific Groups in Gitlab

Note: This example only displays lines that are uncommented. All other lines in the file are commented out.

```
jira:
  url: https://larkin.atlassian.net
  gdpr_active: True
  earliest_issue_dt: 2022-01-01
  issue_download_concurrent_threads: 10
  issue_batch_size: 100
  download_worklogs: True
  download_sprints: True
Git:
  - provider: github
    url: https://api.github.com
    instance_slug: larkin-CB
    creds_envvar_prefix: KDL
    Include_projects:
      - larkin
    include_repos:
      - Kangaroo
      - Koala
      - Penguin
      - Peach
  - provider: gitlab
    url: https://gitlab.com
    instance_slug: larkin-DM
    creds_envvar_prefix: ATL
    include_projects:
      - 9876543
      - 1234567
      - 9245752
    include_repos:
      - 48502
      - 56703
      - 19647
```

Agent usage modes

The Agent has the following usage modes.

Usage mode	Description
download_and_send	Downloads Jira and/or git data from your systems and sends it to Jellyfish
download_only	Downloads Jira and/or git data from your systems
send_only	Sends a previously downloaded dataset to Jellyfish
print_all_jira_fields	Displays the keys and field names of all your custom Jira fields
print_apparently_missing_git_repos	Displays the names and URLs of git repositories that may be missing from Jellyfish
validate	Validates the config.yml file using APIs

Frequently Asked Questions

When setting up an Agent, what are the recommended CPU/memory/storage specs?

- 2 GB of RAM
- 20 GB of storage
- 2 processor cores
- Has network access to the systems you're connecting to
- Docker, preferably a version that supports `--mount` syntax

In terms of storage, this varies a bit as it can be hard to predict how much storage you may need, but we've never seen >20GB. Aside from those, any amount of CPU should work, but if you need a minimum, set it for 2 CPU cores.

If you're using Kubernetes, run the Agent as a Kubernetes job. For additional information, refer to CronJob | Kubernetes.

Note: Jellyfish does not distribute a helm chart for running the Agent as a Kubernetes job.

Does the Agent auto-update? Is there a way to initiate a manual update of the Agent or does it require a deploy of an updated docker image?

Clients can auto-update the Agent before every run. In the cron job/automated run, you can run a docker pull to get the latest version of the Agent.

Is there a communications channel that lets us know when an Agent update is available? Are release log details such as feature additions, security fixes, etc. included?

A member of Jellyfish's Product Success team will reach out with notifications to update the Agent. Logs are captured by the client and can be reviewed at any time.

How should we have the Agent run once it's set up?

Usually utilizing a k8s cronjob, most clients utilize a once or twice-a-day cadence for their Agent configuration. The Agent will exit after it has completed an initial pull. We recommend that you not restart it immediately after completion. The Agent checks the most recent timestamp of commit and only pulls in items after that point. But, if we don't have a most recent timestamp yet (our whole processing pipeline needs to complete), we'll run through all repos again like it never happened.

See also, Running the Agent.

Can we cherry-pick certain Epics to go into Jellyfish via the Agent?

Customers can limit what is sent to Jellyfish via the Agent by modifying the config.yml file.

- To limit project and project categories sent to Jellyfish, refer to Line 56 of the sample config.yml file.
- To limit issues sent to Jellyfish, refer to Line 88 of the sample config.yml file.
- To limit fields sent to Jellyfish, refer to Line 99 of the sample config.yml file.

Is the latest Jellyfish Agent official code base maintained on GitHub?

Yes. https://github.com/Jellyfish-AI/jf_agent

Are contributions on GitHub controlled by the Jellyfish internal team?

Yes

What is the authentication mechanism between the Agent and the Jellyfish server?

API calls between the Agent and the Jellyfish backend are controlled with a token, which your users can invalidate or rotate from within the Jellyfish app. This API is the thing that allows to Agent to get the configuration and status of the data we already have as of what time so that it can pull incremental changes from Jira and git.

That API also returns a pre-signed URL for S3, which is used to transfer the data securely.

Does the Agent connect to anything other than the Jellyfish server?

It connects to Jellyfish and S3.

Is Agent communication bidirectional or unidirectional?

Unidirectional - all traffic is outbound from the Agent.

JELLYFISH

jellyfish.co | LinkedIn Instagram Twitter Youtube