
Plynx Documentation

Release 1.2.3

Plynx

Jun 04, 2023

Contents

1	Content	3
1.1	Overview	3
1.1.1	Demo	4
1.1.2	Open source	4
1.1.3	Other links	4
1.2	Get started	4
1.2.1	Run local cluster	4
1.2.2	Other <i>make</i> commands	5
1.3	Concepts	5
1.3.1	Graph	5
1.3.2	Operation	6
1.3.3	Creating operations	8
1.4	Configuration	10
1.4.1	Executors	10
1.4.2	Storage	10
1.4.3	Auth	11
1.4.4	Web	11
1.4.5	Plugins	11
1.4.5.1	Workflows	13
1.4.5.2	Hubs	13
1.4.5.3	Operations	14
1.4.5.4	Resources	14
1.5	Plugins	14
1.5.1	Executors	15
1.5.2	Resources	16
1.5.3	Hubs	17
1.6	REST API	17
1.6.1	Authentication	18
1.6.1.1	Response Structure	18
1.6.2	List multiple Graphs	18
1.6.2.1	Request Structure	18
1.6.2.2	Response Structure	19
1.6.2.3	Example	19
1.6.3	Get single Graph	19
1.6.3.1	Response Structure	19
1.6.3.2	Example	19

1.6.4	Post Graph	19
1.6.4.1	Data	20
1.6.4.2	Actions	20
1.6.4.3	Response Structure	20
1.6.5	Single action endpoints	20
1.6.5.1	Example	21
1.6.6	List multiple Nodes	22
1.6.6.1	Request Structure	22
1.6.6.2	Response Structure	22
1.6.6.3	Example	22
1.6.7	Get single Node	22
1.6.7.1	Response Structure	23
1.6.7.2	Example	23
1.6.8	Post Node	23
1.6.8.1	Data	23
1.6.8.2	Actions	24
1.6.8.3	Response Structure	24
1.6.9	Upload File	24
1.6.9.1	Example	25
1.6.10	Modify existing Graphs	25
1.6.10.1	Example	25
1.6.11	Working with a single Resource	26
1.6.12	Get state of the Master	26
1.6.12.1	Example	27
1.7	Internal data structures	27
1.7.1	Graph	27
1.7.1.1	Graph Running Status	28
1.7.2	Node	28
1.7.2.1	Input	28
1.7.2.2	Input Value	29
1.7.2.3	Parameter	29
1.7.2.4	Output	29
1.7.2.5	Log	29
1.7.2.6	Node Running Status	29
1.7.2.7	Node Status	30
1.8	Kubernetes deployment	30
1.8.1	Step 0: Create a cluster	30
1.8.2	Step 1: Create service account credentials	31
1.8.3	Step 2: Create secret key	31
1.8.4	Step 3: Create MongoDB pod	31
1.8.5	Step 4: PLynx pods	32
1.8.6	Step 5: Init users	32
1.8.7	Step 6: Try PLynx	32
1.9	Frequently Asked Questions	33
1.9.1	Who is it for?	33
1.9.2	Can I run it without docker?	33
1.9.3	How is it different from Airflow, Kubeflow and other platforms?	33
1.9.4	Is it a no-code platform?	33
1.9.5	How can I install additional packages?	33
1.9.6	How to contact us?	34
1.10	License	34
1.11	API Reference	37
1.11.1	plynx	37
1.11.1.1	Subpackages	37

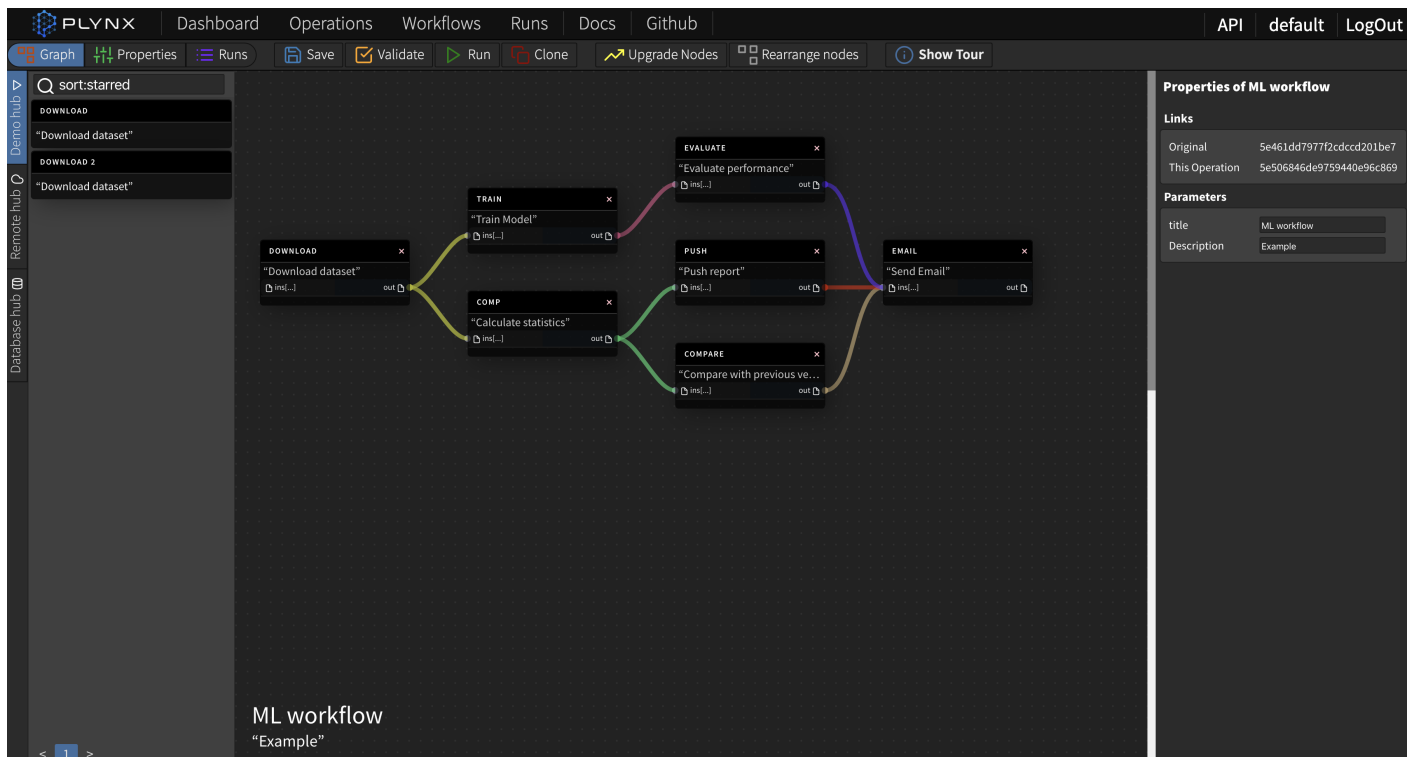
1.11.1.2 Package Contents	87
Python Module Index	89
Index	91

PLynx is a domain agnostic platform for managing reproducible experiments and data-oriented workflows.

CHAPTER 1

Content

1.1 Overview



PLynx is a high level domain agnostic orchestration and computation platform. It is not constrained by a particular application and can be extended to custom data oriented workflows and experiments.

The platform abstracts users from the engineering and organizational complexities, such as data access, containerization, remote computation, automatic failover, progress monitoring, and other advanced computer science concepts.

The core principles of Plynx include:

- *Domain agnostic.* Plynx orchestrates multiple services with higher level APIs, such as various cloud services, version control repositories, databases and clusters.
- *Flexible experimentation.* Users are not limited by pre-defined experiment structure or approved operations. They are encouraged to reuse existing solutions, but if necessary they can introduce their solutions.
- *Reproducible experiments.* Results of each of the experiments are accessible and reusable. Past experiments and ideas can be reused anytime by anyone without technical challenges.
- *Parallel execution.* It is possible to conduct multiple experiments simultaneously.
- *Caching.* Results of previously executed Operations and subgraphs will be reused.
- *Monitoring.* Plynx abstracts orchestration, visualization, workflow version control, sharing the results and others.

1.1.1 Demo

Demo is available at <https://plynx.com>.

1.1.2 Open source

Plynx is licensed under Apache 2.0. Source code is available at [Github](#).

1.1.3 Other links

Please refer to an article in [Medium](#) about the goals of the project.

1.2 Get started

1.2.1 Run local cluster

Make sure you install docker first. [Get started with Docker](#)

tl;dr

```
git clone https://github.com/khaxis/plynx.git    # Clone the repo
cd plynx
cp template_config.yaml config.yaml              # Make a copy of a config
make up                                           # to start production services
```

Then go to <http://localhost:3001>

It will start the following services:

- MongoDB instance
- Plynx User Interface
- API service
- DAG worker

- Several Python and Bash workers (5 by default)

Run `make down` to stop the services.

1.2.2 Other *make* commands

- `make build` - build all docker images.
- `make run_tests` - build docker images and run the tests.
- `make up` - run the services locally.
- `make down` - shut down services.
- `make dev` - run developer version of Plynx.

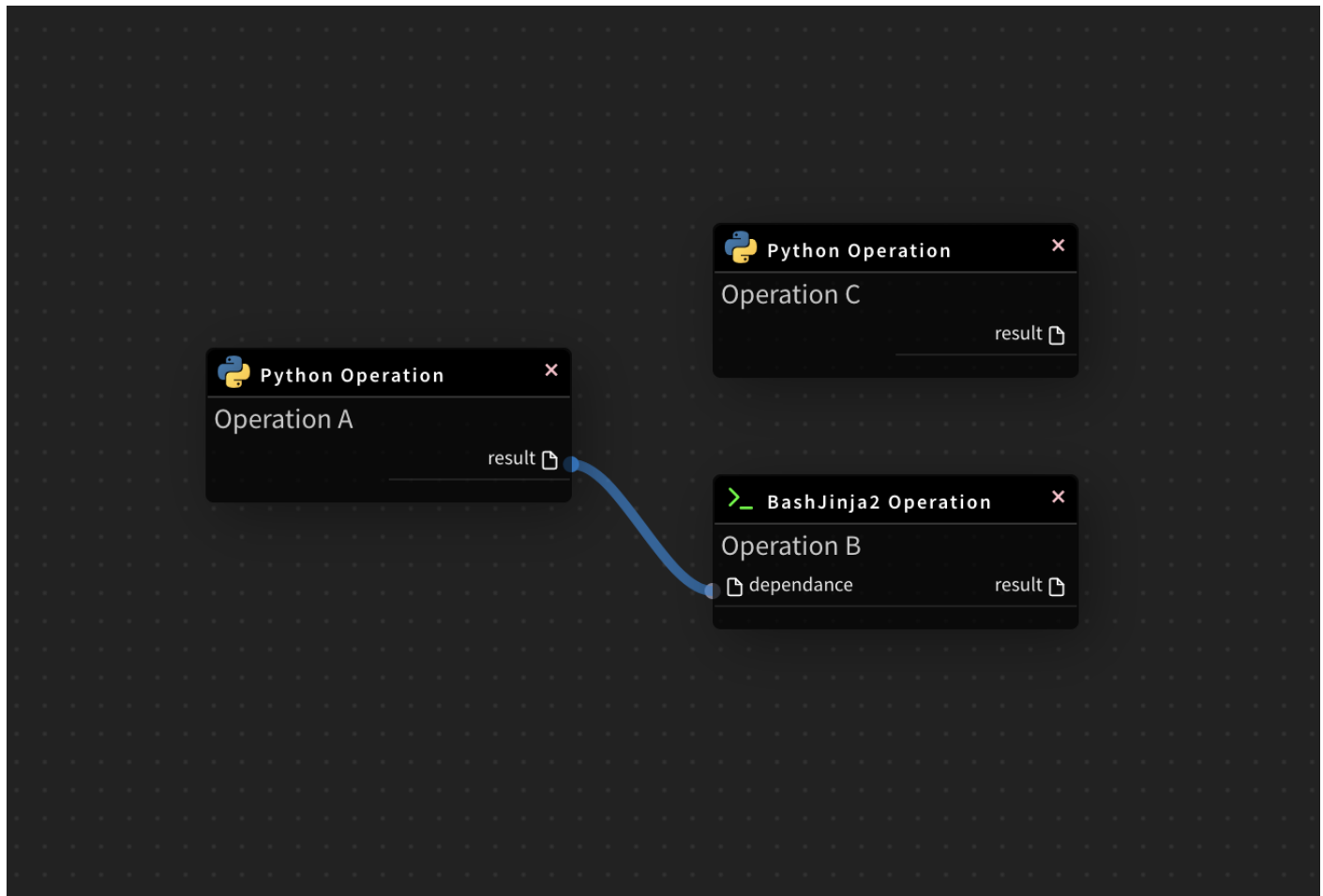
1.3 Concepts

Plynx is a domain agnostic platform for managing reproducible experiments and data-oriented workflows.

1.3.1 Graph

In Plynx, each experiment is represented as Directed Acyclic Graph or simply Graph – a collection of all the nodes you want to run, organized in a way that reflects their relationships and dependencies.

For example, a simple graph could consist of three operations: A, B, and C. It could say that A has to run successfully and produce a resource before B can run, because B depends on it. But C can run anytime. In other words graph describes how you want to carry out your workflow.



Notice that we haven't said anything about what we actually want to do! A, B, and C could be anything:

- A prepares data for B to analyze while C sends an email.
- A monitors your location so B can open your garage door while C turns on your house lights.

The important thing is that the graph isn't concerned with what its constituent nodes do; its job is to make sure that whatever they do happens in the right order.

Unlike existing solutions (such as Apache Airflow), Plynx makes no assumption about the structure of its graphs. Each graph can be created dynamically using UI or APIs. In data science it is common to try new features or ideas with custom layout. Multiple ideas might fail before you find the promising one. Plynx makes it easy to create new experiments based on existing graphs and try them simultaneously and reproduce results if needed.

1.3.2 Operation

While graphs describe how to run an experiment, operations determine what actually gets done.

Operation is a building block of the graphs. They describe a single executable task in a workflow. Operations are usually (recommended) atomic, meaning they share only input and output resources with any other operations.

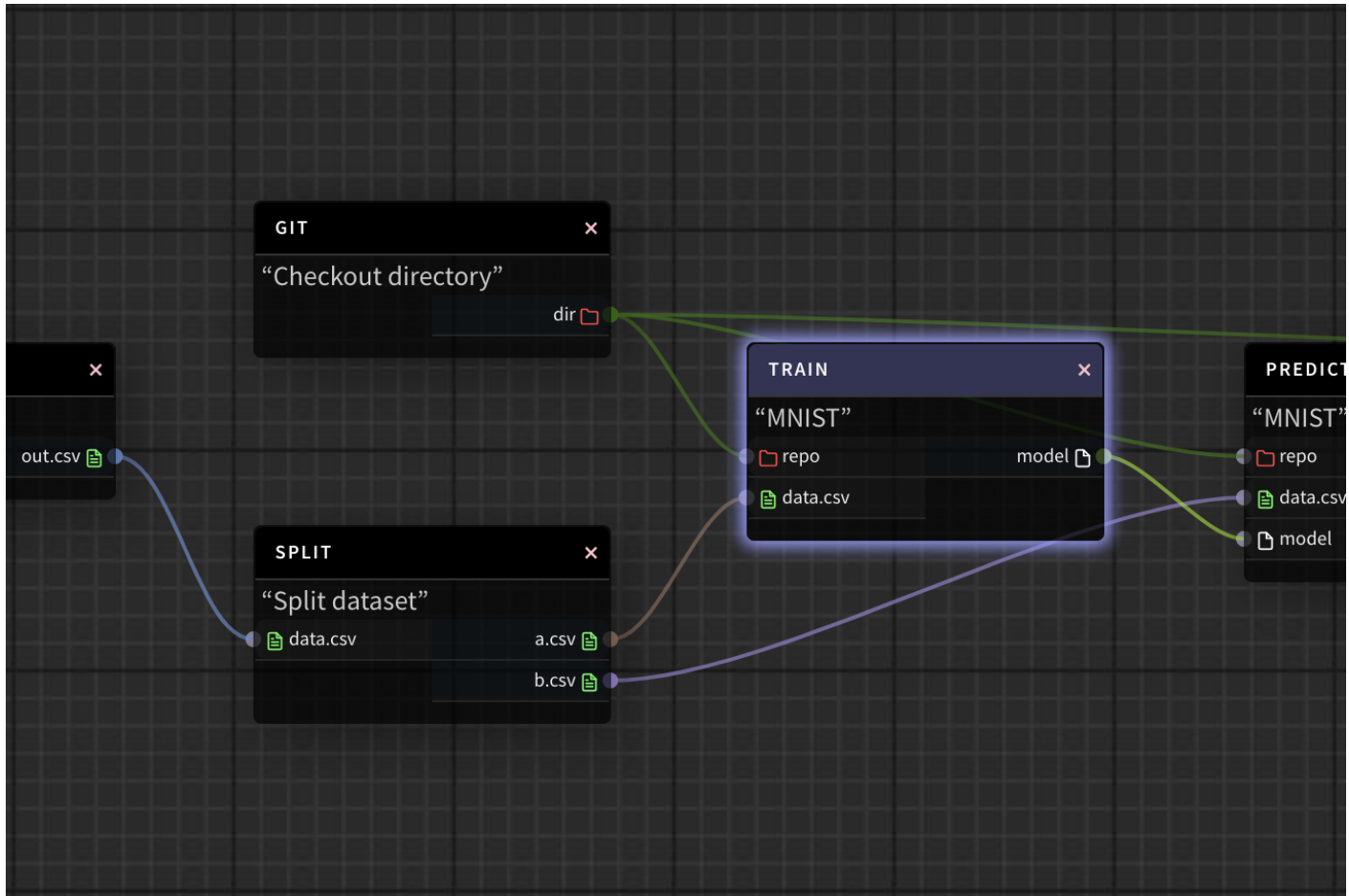
Plynx is using multiple types of Operations that can be customized by plugins. Here are some of them.

- *Python Operation* executes code in python.
- *BashJinja2 Operation* uses jinja2 templates to execute bash script.

- *Composite Operation* consists of multiple other operations. It can be considered as a sub-graph.

The graph will make sure that operations run in the correct certain order; other than those dependencies, operations generally run independently. In fact, they may run on two completely different machines.

This is a subtle but very important point: in general, operation should be atomic. If there is an intermediate resource an Operator can produce which can be used by other consumer, such as transformed data or subset, you should consider splitting the operation. It encourages users to create their workflows in a more modular and parameterized way reusing existing solutions. This way they don't reinvent existing solutions multiple times and can use advantages of cached results and distributed computation.



The graph above is a part of a machine learning pipeline. Plynx will execute operations in the order defined by the graph. In the example above, *Train* operation requires two *Resources*: *repo* and *data.csv*. As soon as these resources are available, Plynx worker will pick this job up and execute it. In this sense Plynx is very similar to *Makefiles*.

Resource preparation and execution is defined by internal Plynx class called `BaseNode`. Currently it includes the following ones:

<code>base_node_name</code>	Description
<code>file</code>	It is a dummy <code>BaseNode</code> . The File gets never executed. Instead of that it has a single output called <i>out</i> which is known before execution.
<code>bash_jinja2</code>	It executes a custom bash command. Users specify external resources and parameters with Jinja2 templating language. See examples Creating operations .
<code>python</code>	Custom python script will be specified by this <code>BaseNode</code> . See examples Creating operations .

1.3.3 Creating operations

Users are responsible for defining operations. Say we have a git repository where we keep scripts for each step for machine learning pipeline. *Git - checkout directory* is an operation defined by a user. Given a link to a repository and commit hash the operation clones the repository and creates a new resource in Plynx. The resource is called `dir` and has a type *Directory*. The directory might contain multiple scripts and can be reused by other operations.

Custom properties		Internal properties		✖ Deprecate <> Preview 🔗 Clone	
Title	Git	Node Status	READY		
Description	Checkout directory	Parent Node	5d38e649e2e290722ae8f224		
Base Node	bash_jinja2	Successor	null		
		Created	2019-07-24 23:29:51.714000		
		Updated	2019-07-24 23:29:51.714000		

Inputs	Parameters	Outputs
Name: <code>cmd</code> Type: <code>Code</code>	Value: <code>sh</code> <pre> 1 set -e 2 3 # clone repo 4 export DIRECTORY=directory 5 git clone {{ param['repo'] }} \$DIRECTORY 6 cd \$DIRECTORY 7 8 # reset to custom commit hash 9 git reset --hard {{ param['commit'] }} 10 11 # build using custom build command 12 cp -r . {{ output.dir }} 13 </pre>	Name: <code>dir</code> Type: <code>Directory</code>
Name: <code>cacheable</code> Type: <code>Boolean</code>	Value: ☒ True	
Name: <code>_timeout</code> Type: <code>Integer</code> Widget: ☐ Public	Value: 600	
Name: <code>repo</code> Type: <code>String</code> Widget: ☒ Public Repo URL	Value: <code>https://github.com/khaxis/plynx-mnist-demo</code>	
Name: <code>commit</code> Type: <code>String</code> Widget: ☒ Public Commit #	Value: <code>058fb559d76c394f77494d4c7e276b17120689b9</code>	

The script that defines *Git - checkout directory* operation can be found in a system parameter `cmd`:

```

set -e

# clone repo
export DIRECTORY=directory
git clone {{ params['repo'] }} $DIRECTORY
cd $DIRECTORY

# reset to custom commit hash
git reset --hard {{ params['commit'] }}

# build using custom build command

```

(continues on next page)

(continued from previous page)

```
cp -r . {{ outputs.dir }}
```

Before executing the script, Plynx worker will prepare inputs: it will download and preprocess inputs and create empty outputs. The worker will create an empty directory. The path to this directory is not known in advance: in order to avoid race condition on the filesystem each process will be working with temporary path. You can find the exact path using `{{ inputs.* }}` or `{{ outputs.* }}` placeholders. In *git* example you it would be `{{ outputs.dir }}`.

The screenshot displays the Plynx node configuration interface. It includes tabs for 'Custom properties' and 'Internal properties'. The 'Custom properties' tab shows the node's title, description, and base node. The 'Internal properties' tab shows the node's status, parent node, successor, and timestamps. The 'Inputs' section shows the input data source and count. The 'Parameters' section shows a code editor with a Python script for splitting data. The 'Outputs' section shows the output data sources and types.

Similarly operation can be defined in python. Instead of *jinja2* templates use python variables `inputs`, `outputs`, and `params`.

```
import random

def split(inputs, output_a, output_b, sample_rate, seed):
    random.seed(seed)
    with open(output_a, 'w') as fa, open(output_b, 'w') as fb:
        for input_filename in inputs:
            with open(input_filename, 'r') as fi:
                for line in fi:
                    if random.random() < sample_rate:
                        fa.write(line)
                    else:
                        fb.write(line)
```

(continues on next page)

(continued from previous page)

```
split(
    inputs=inputs['data.csv'],
    output_a=outputs['a.csv'],
    output_b=outputs['b.csv'],
    seed=int(params['seed']),
    sample_rate=float(params['rate']),
)
```

1.4 Configuration

Configuration file is a way to customize and extend PLYnx for your needs. It defines structure of the platform, declares plugins and their interaction.

An example of a config file can be found here: [config.yaml](#)

1.4.1 Executors

PLYnx is using MongoDB to store Workflows, Operations, Runs and other metadata. API and Workers need to have access to the database to work properly.

```
mongodb:
  user: <username>
  password: <password>
  host: <server ip>
  port: <server port>
```

1.4.2 Storage

All of the artifacts will be stored as files. Your Resource plugins *Resources* define the way how to handle them.

```
storage:
  scheme: <scheme_label>
  prefix: <prefix>
```

Here are possible schemas that works as a driver to a file storage.

Scheme	Prefix example	Description
file	/data/ resources/	Using a local directory as a file storage. Note the the file system should be accessible by API and all of the workers. This can be done by running everything on a single machine or mounting a persistent storage.
gs	gs:///plynx-resources/	Google Cloud Storage driver.
s3	s3:///plynx-resources/	AWS s3 driver.

1.4.3 Auth

PLynx can run in multiuser mode if you provide it with a secret key. It is used to generate authentication tokens.

Note you may also use `--secret-key` argument running API server in order to avoid keeping the key in a file for security concerns.

```
auth:
  secret_key: <secret key>
```

1.4.4 Web

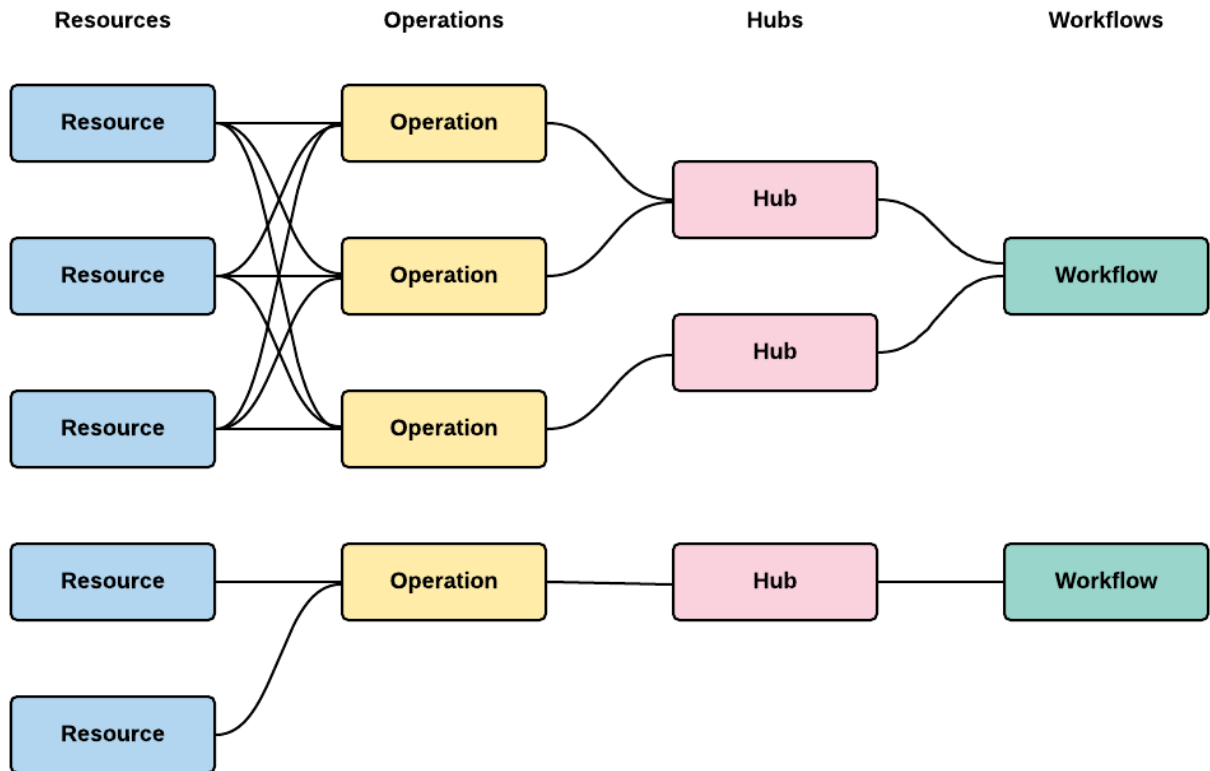
This part of a configuration file is used by the API server. It customizes flask web service.

```
web:
  host: <host>
  port: <port>
  endpoint: <endpoint>
  debug: <debug mode>
```

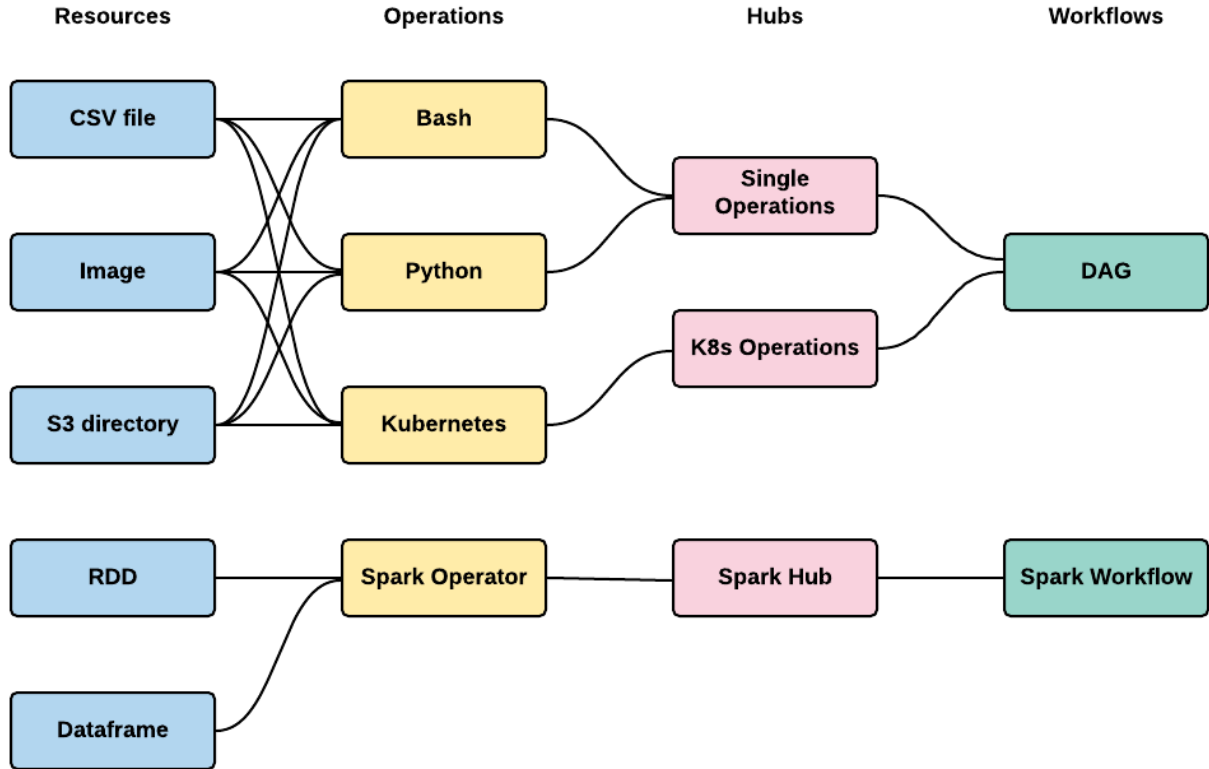
1.4.5 Plugins

The architecture of Plynx supports various use cases. It does not rely on a single backend or workflow. Plynx is basically a graph editor with plugins that provide with backend computation.

Simple structure of plugins is show below.



Please look at an example below. This architecture supports two kinds of Workflows with their own backend and Operations. It can be all configured in `config.yaml` without any change to Plynx. Using `config.yaml` allows to set up Plynx for your use case without making changes to the platform to avoid forking and using outdated version of Plynx.



1.4.5.1 Workflows

There can be multiple types of Workflows that share some Hubs or no Hubs at all. Workflow backend is provided by *Executors*.

Examples: - DAG executor that runs operations one by one or in parallel based on the order given by graph structure. - Spark executor that materializes graph into spark code. - Image processing pipeline that applies filters and transformations in given order.

```
workflows:
- kind: <unique identifier>
  title: <readable title>
  executor: <executor class>
  hubs:
    - <list of hub identifiers>
```

1.4.5.2 Hubs

Main reason of using Hubs is to organize Operations in a useful and convenient way. Hubs are using backend from *Hubs*.

Examples: - Database hub searches all of the Operations from the database. - Static hub serves a static list of Operations. - PLYnx supports hierarchical structure of a catalog of Operations using Groups. - Using external sources that open new Operations libraries or serve additional features such as version control.

```
hubs:
- kind: <unique identifier>
  title: <readable title>
  icon: <icon>
  cls: <hub class>
  args:
    <additional arguments>
```

1.4.5.3 Operations

Operations form building blocks for Workflows. They can be either atomic, i.e. represent a single script, or they can consist of other Operations. Operation backend is provided by [Executors](#).

Examples: - Python script. - Bash script. - Same as Python or Bash, but using its own environment, such as additional API or SDK or custom hardware.

```
operations:
- kind: <unique identifier>
  title: <readable title>
  executor: <executor class>
  icon: <icon>
  color: <color>
  resources:
    <list of resource identifiers>
```

1.4.5.4 Resources

Resources are an abstraction for working with artifacts. Each artifact in PLYND is a file. Resources define some standardized interface to work them. Resources backend is provided by [Resources](#).

Examples: - Simple file. - Directory. It will be stored as an archive in [Storage](#). Directory resource will take care of unzipping it before starting an Operation and will take of it when it successfully finishes. - Executable. This is a file with unix flag +x set.

```
resources:
- kind: <unique identifier>
  title: <readable title>
  cls: <resource class>
  icon: <icon>
  color: <color>
```

1.5 Plugins

Plynx offers a generic interface for working with custom infrastructure and services. Different organizations have different stacks and different needs. Using Plynx plugins can be a way for companies to customize their Plynx installation to reflect their ecosystem.

Plugins can be used as an easy way to write, share and activate new sets of features.

There's also a need for a set of more complex applications to interact with different flavors of data and backend infrastructure. Essentially Plynx is a graph editor with computational backend provided by plugins and configuration.

Examples:

- Inputs and Outputs might have explicitly different types. They can be simple files or references to S3 / Big Query table / HDFS path etc.
- Users have an option to customize pre or post-processing as well as preview options.
- Operations can be execute commands in multiple languages or in custom environments.
- Some applications require executing Operations one by one, others transform DAGs into AWS Step functions or Argo utilizing 3rd party backend.
- Catalog and categorization of Operations can be supported bu Hubs.
- ...

1.5.1 Executors

Executors are the mechanism by which job instances get run.

PLynx has support for various executors. Some of them execute a single job such as python or bash script. Others are working with DAG structure.

PLynx can be customized by additional executors.

```
import plynx.base.executor

class CustomExecutor(plynx.base.executor.BaseExecutor):
    """ Custom Executor.

    Args:
        node_dict (dict)

    """

    def __init__(self, node_dict):
        super(DAG, self).__init__(node_dict)

        # Initialization

    @classmethod
    def get_default_node(cls, is_workflow):
        """
        You may modify a node by additional parameters.
        """
        node = super().get_default_node(is_workflow)

        # customize your default node here

        return node

    def run(self):
        """
        Worker will execute this function.
        """

    def kill(self):
        """
        Worker will call this function if parent executor or a user canceled the
        ↪ process.
```

(continues on next page)

(continued from previous page)

```
"""

def validate(self):
    """
    Validate Node.
    """
```

1.5.2 Resources

PLynx explicitly define artifacts: Inputs, Outputs, and Logs. The mechanism to handle custom artifacts is supported by Resource plugins.

```
import plynx.base.resource

class CustomResource(plynx.base.resource.BaseResource):
    @staticmethod
    def prepare_input(filename, preview):
        """
        Prepare resource before execution.
        """
        if preview:
            return {NodeResources.INPUT: filename}
        # Customize preprocessing here
        return {NodeResources.INPUT: filename}

    @staticmethod
    def prepare_output(filename, preview):
        """
        Prepare output resource before execution.

        For example create a directory or an empty file.
        """
        if preview:
            return {NodeResources.OUTPUT: filename}
        # Customize preprocessing here
        return {NodeResources.OUTPUT: filename}

    @staticmethod
    def postprocess_output(filename):
        """
        Process output after execution.

        For example compress a file or compute extra statistics.
        """
        return filename

    @classmethod
    def preview(cls, preview_object):
        """
        Redefine preview function.

        For example display text content or <img>
        """
```

(continues on next page)

(continued from previous page)

```
return '<pre>{}</pre>'.format(content)
```

1.5.3 Hubs

Hubs let users to organize Operations in the editor and use additional sources.

```
import plynx.base.hub

class CustomHub(plynx.base.hub.BaseHub):
    def __init__(self, **argv):
        super(CollectionHub, self).__init__()

        # use arguments to customize the hub

    def search(self, query):
        """
        Customize search.
        """
```

Warning: API docs are outdated. Please contact us for details.

1.6 REST API

The Plynx REST API allows you to perform most of the actions with Graphs, Operations and Files. The API is hosted under the `/plynx/api/v0` route on the Plynx server. For example, to list latest Graphs on backend server hosted at `http://localhost:5005`, make GET request: `http://localhost:5005/plynx/api/v0/graphs`.

Table of Contents

- *Authentication*
- *List multiple Graphs*
- *Get single Graph*
- *Post Graph*
- *Single action endpoints*
- *List multiple Nodes*
- *Get single Node*
- *Post Node*
- *Upload File*
- *Modify existing Graphs*
- *Working with a single Resource*
- *Get state of the Master*

1.6.1 Authentication

User session should start with this endpoint. If a user is verified, response will contain access and refresh tokens.

Endpoint	HTTP Method
<code>plynx/api/v0/token</code>	GET

1.6.1.1 Response Structure

Field Name	Type	Description
<code>status</code>	STRING	Status of the query. One of the following: <code>success</code> or <code>failed</code>
<code>message</code>	STRING	Extended status.
<code>access_token</code>	STRING	Short-term token.
<code>refresh_token</code>	STRING	Long-term token.

1.6.2 List multiple Graphs

Endpoint	HTTP Method
<code>plynx/api/v0/search_graphs</code>	POST

Blank line required after table.

Get a list of Graphs. You can specify search parameters as a request body, for example:

```
{
  "per_page": 20,
  "offset": 0,
  "search": "author:default "
}
```

1.6.2.1 Request Structure

Parameter name	Type	Default value	Description
<code>search</code>	STRING	<code>" "</code>	Search string. See <code>plynx-internal-search-string</code> for more details.
<code>per_page</code>	INTEGER	20	Number of instances returned by the query.
<code>offset</code>	INTEGER	0	Number of instances to skip.
<code>status</code>	STRING or LIST	<code>[]</code>	List of statuses. See <i>Graph Running Status</i> for more details.

1.6.2.2 Response Structure

Field Name	Type	Description
items	An array of <i>Graph</i>	All experiments.
total_count	INTEGER	Total number of graphs that meet the query.
status	STRING	Status of the query. One of the following: success or failed

1.6.2.3 Example

```
curl -X POST \
  'http://localhost:5005/plynx/api/v0/search_graphs' \
  -u default: -H "Content-Type: application/json" \
  -d '{"per_page":1, "search":"author:default"}'
```

1.6.3 Get single Graph

Endpoint	HTTP Method
plynx/api/v0/graphs/{graph_id}	GET

Get a single Graph in *Graph* format.

Parameter `graph_id` is required.

When `graph_id == "new"` (i.e. `curl 'http://localhost:5005/plynx/api/v0/graphs/new' -u default:`) Plynx backend will generate a default empty Graph. Please note this new Graph will not be saved in the database. Use POST request instead `plynx-rest-post_graph`:

1.6.3.1 Response Structure

Field Name	Type	Description
data	<i>Graph</i>	Graph object.
re-sources_dict	plynx-internal-resources_dict	Dictionary of available resources types that come as plugins.
status	STRING	Status of the query. One of the following: success or failed

1.6.3.2 Example

```
curl 'http://localhost:5005/plynx/api/v0/graphs/5d1b8469705c1865e288a664' -u default:
```

1.6.4 Post Graph

Endpoint	HTTP Method
plynx/api/v0/graphs	POST

This endpoint covers multiple actions with a Graph, such as saving, approving, generating code, etc. A single request can contain a sequence of actions that will be applied in the same order.

Note that some of the actions that require a change in the database, are not always permitted. For example when the user is not the original author of the Graph. In this case the Graph is considered as `read only`.

1.6.4.1 Data

Parameter name	Type	Description
graph	<i>Graph</i>	Graph object.
action	LIST of STRING	List of actions. See <i>Actions</i> for more details.

1.6.4.2 Actions

Action Name	Description	Permission limitations	Extra fields in response
SAVE	Save the graph. If the Graph with the same Id does not exist, it will be created.	Author must match the current user	
APPROVE	Save the graph and execute it if it passes validation.	Author must match the current user	validation_error
VALIDATE	Check if the Graph passes validation, i.e. cycles detected, invalid inputs, etc.	Any User	validation_error
REAR-RANGE	Rearrange Nodes based on topology of the Graph.	Any User	
UP-GRADE_NODES	Replace outdated nodes with new versions	Any User	upgraded_nodes_count
CANCEL	Cancel currently running Graph.	Author must match the current user	
GENERATE_CODE	Generate python API code that can recreate the same graph.	Any User	code
CLONE	Clone the graph and save it.	Any User	new_graph_id

1.6.4.3 Response Structure

Field Name	Type	Description
graph	<i>Graph</i>	Graph object.
url	STRING	URL.
message	STRING	Dictionary of available resources types that come as plugins.
status	STRING	Status of the query. One of the following: <code>success</code> or <code>failed</code> or <code>validation_failed</code>
validation_error (extra)	An array of plynx-internal-validation_error	If errors found on validation step.
up-graded_nodes_count (extra)	INTEGER	Dictionary of available resources types that come as plugins.
code (extra)	STRING	Resulting code

1.6.5 Single action endpoints

Similarly to *Actions*, you can perform actions with existing Graphs. These POST-requests do not require json data. Backend will use existing Graph instead.

Endpoint	HTTP Method	Data
plynx/api/v0/graphs/{graph_id}/approve	POST	None
plynx/api/v0/graphs/{graph_id}/validate	POST	None
plynx/api/v0/graphs/{graph_id}/rearrange	POST	None
plynx/api/v0/graphs/{graph_id}/upgrade_nodes	POST	None
plynx/api/v0/graphs/{graph_id}/cancel	POST	None
plynx/api/v0/graphs/{graph_id}/generate_code	POST	None
plynx/api/v0/graphs/{graph_id}/clone	POST	None

Additional PATCH endpoint is available to update the Graph.

Endpoint	HTTP Method	Data
plynx/api/v0/graphs/{graph_id}/update	PATCH	JSON, required

1.6.5.1 Example

```
# Clone existing Graph
curl -X POST \
  'http://localhost:5005/plynx/api/v0/graphs/5d1b8469705c1865e288a664/clone' \
  -u default:
# {"status": "SUCCESS", "message": "Actions completed with Graph(_
↪ id='5d1b8469705c1865e288a664')", "graph": {"_id": "5d291e57713b286094d4ad85", "title
↪ ": "hello world", "description": "Description", "graph_running_status": "CREATED",
↪ "author": "5d0686aa52691468eaef391c", "nodes": [{"_id": "5d27e3bd0f432b5e3693314c",
↪ "title": "Sum", "description": "Sum values", "base_node_name": "python", "parent_
↪ node": "5d27b8dd50e56dbbce063449", "successor_node": null, "inputs": [{"name":
↪ "input", "file_types": ["file"], "values": [], "min_count": 1, "max_count": -1}],
↪ "outputs": [{"name": "output", "file_type": "file", "resource_id": null}],
↪ "parameters": [{"name": "cmd", "parameter_type": "code", "value": {"value": "s =
↪ 0\nfor filename in input[\"input\"]:\n    with open(filename) as fi:\n        s +=
↪ sum([int(line) for line in fi])\nwith open(output[\"output\"], \"w\") as fo:\n
↪ fo.write(\"{}\n\".format(s))\n", "mode": "python", "mutable_type": false,
↪ "removable": false, "publicable": false, "widget": null}, {"name": "cacheable",
↪ "parameter_type": "bool", "value": true, "mutable_type": false, "removable": false,
↪ "publicable": false, "widget": null}], "logs": [{"name": "stderr", "file_type":
↪ "file", "resource_id": null}, {"name": "stdout", "file_type": "file", "resource_id
↪ ": null}, {"name": "worker", "file_type": "file", "resource_id": null}], "node_
↪ running_status": "CREATED", "node_status": "READY", "cache_url": "", "x": 190, "y":
↪ 143, "author": "5d0686aa52691468eaef391c", "starred": false}}], "url": "http://
↪ localhost:3001/graphs/5d291e57713b286094d4ad85", "new_graph_id":
↪ "5d291e57713b286094d4ad85"}
```

```
# Change Title and Description
# Note "new_graph_id": "5d291e57713b286094d4ad85"
curl -X PATCH \
  'http://localhost:5005/plynx/api/v0/graphs/5d1b8469705c1865e288a664/update' \
  -u default: -H "Content-Type: application/json" \
  -d '{"title": "Custom title", "description": "Custom Description"}'
```

```
# Execute the Graph:
curl -X POST \
  'http://localhost:5005/plynx/api/v0/graphs/5d1b8469705c1865e288a664/approve' \
  -u default:
```

1.6.6 List multiple Nodes

Note Files and Operations internally are represented as Nodes.

Endpoint	HTTP Method
<code>plynx/api/v0/search_nodes</code>	POST

Get a list of Nodes. You can specify search parameters as a request body, for example:

```
{
  "per_page": 20,
  "offset": 0,
  "search": "author:default "
}
```

1.6.6.1 Request Structure

Parameter name	Type	Default value	Description
search	STRING	" "	Search string. See <code>plynx-internal-search-string</code> for more details.
per_page	INTEGER	20	Number of instances returned by the query.
offset	INTEGER	0	Number of instances to skip.
status	STRING or LIST	[]	List of statuses. See Node Status for more details.
base_node_names	LIST of <code>plynx-internal-base_node</code>	[]	List of base nodes. See <code>plynx-internal-base_node</code> for more details.

1.6.6.2 Response Structure

Field Name	Type	Description
items	An array of Node	Nodes (Operations and Files)
re-sources_dict	An array of <code>plynx-internal-resources_dict</code>	List of resources available in the platform.
total_count	INTEGER	Total number of nodes that meet the query.
status	STRING	Status of the query. One of the following: <code>success</code> or <code>failed</code>

1.6.6.3 Example

```
curl -X POST \
  'http://localhost:5005/plynx/api/v0/search_nodes' \
  -u default: -H "Content-Type: application/json" \
  -d '{"per_page":1, "search":"author:default"}'
```

1.6.7 Get single Node

Endpoint	HTTP Method
<code>plynx/api/v0/nodes/{node_id}</code>	GET

Get a single Graph in *Node* format.

There are special cases when *node_id* is *base_node_name*, i.e. `curl 'http://localhost:5005/plynx/api/v0/nodes/python'` or `curl 'http://localhost:5005/plynx/api/v0/nodes/bash_jinja2'`. Backend will generate a default Operation.

1.6.7.1 Response Structure

Field Name	Type	Description
data	<i>Node</i>	Node object.
re-sources_dict	plynx-internal-resources_dict	Dictionary of available resources types that come as plugins.
status	STRING	Status of the query. One of the following: success or failed

1.6.7.2 Example

```
curl 'http://localhost:5005/plynx/api/v0/nodes/5d27b8dd50e56dbbce063449' -u default:
```

1.6.8 Post Node

Endpoint	HTTP Method
plynx/api/v0/nodes	POST

This endpoint covers multiple actions with a Node, such as saving, approving, deprecating, etc.

Note that some of the actions that require a change in the database, are not always permitted. For example when the user is not the original author of the Node. In this case the Node is considered as `read only`.

1.6.8.1 Data

Parameter name	Type	Description
node	<i>Node</i>	Node object.
action	STRING	List of actions. See <i>Actions</i> for more details.

1.6.8.2 Actions

Action Name	Description	Permission limitations	Extra fields in response
SAVE	Save the Node. If the Node with the same Id does not exist, it will be created.	Author must match the current user	
APPROVE	Save the Node and make accessible in Graphs if it passes validation.	Author must match the current user	validation_error
VALIDATE	Check if the Node passes validation, i.e. incorrect parameter values.	Any User	validation_error
DEPRECATE	Deprecate the Node. User will still be able to use it.	Author must match the current user	
MANDATORY_DEPRECATE	Deprecate the Node mandatory. Users will no longer be able to use it.	Author must match the current user	validation_error
PREVIEW_CMD	Preview exec script.	Any User	validation_error

1.6.8.3 Response Structure

Field Name	Type	Description
node	<i>Node</i>	Node object.
url	STRING	URL.
message	STRING	Extended status.
status	STRING	Status of the query. One of the following: success or failed or validation_failed
validation_error (extra)	An array of plynx-internal-validation_error	If errors found on validation step.
preview_text (extra)	STRING	Resulting code.

1.6.9 Upload File

This endpoint will create a new Node with type *File*. If you work with large files it is recommended to use an external file storage and Operation that downloads the file (i.e. S3).

Endpoint	HTTP Method	Data
plynx/api/v0/upload_file	POST or PUT	Forms, required

Form	Description
data	Binary data of the file.
title	Title of the file
description	Description of the file
file_type	Type, i.e. <i>file</i> , <i>csv</i> , <i>image</i> , etc.

1.6.9.1 Example

```
curl \
-X POST \
'http://localhost:5005/plynx/api/v0/upload_file' \
-u default: \
-H "Content-Type: multipart/form-data" \
-F data=@/tmp/a.csv \
-F title=report \
-F description=2019 \
-F file_type=csv \
-F node_kind=basic-file
```

1.6.10 Modify existing Graphs

Endpoint	HTTP Method	Data
plynx/api/v0/graphs/{graph_id}/nodes/list_nodes	GET	<i>None</i>
plynx/api/v0/graphs/{graph_id}/nodes/insert_node	POST	<i>node_id: required.</i> <i>x: optional. Default: 0.</i> <i>y: optional. Default: 0.</i>
plynx/api/v0/graphs/{graph_id}/nodes/remove_node	POST	<i>node_id: required.</i>
plynx/api/v0/graphs/{graph_id}/nodes/create_link	POST	<i>from: required. Type: Object. Output node description.</i> <i>from.node_id: required.</i> <i>from.resource: required. Name of the Output</i> <i>to: required. Type: Object. Input node description.</i> <i>to.node_id: required.</i> <i>to.resource: required. Name of the Input</i>
plynx/api/v0/graphs/{graph_id}/nodes/remove_link	POST	<i>from: required. Type: Object. Output node description.</i> <i>from.node_id: required.</i> <i>from.resource: required. Name of the Output</i> <i>to: required. Type: Object. Input node description.</i> <i>to.node_id: required.</i> <i>to.resource: required. Name of the Input</i>
plynx/api/v0/graphs/{graph_id}/nodes/change_parameter	POST	<i>node_id: required.</i> <i>parameter_name: required.</i> <i>parameter_value: required.</i>

1.6.10.1 Example

```
curl -X POST 'http://localhost:5005/plynx/api/v0/graphs/5d292406713b286094d4ad87/
↪nodes/insert_node' \
  -u default: -H "Content-Type: application/json" \
  -d '{"node_id": "5d2d4b1dc36682386f559eae", "x": 100, "y": 100}'

curl -X POST 'http://localhost:5005/plynx/api/v0/graphs/5d292406713b286094d4ad87/
↪nodes/remove_node' \
  -u default: -H "Content-Type: application/json" \
  -d '{"node_id": "5d27e3bd0f432b5e3693314c"}'

curl -X POST 'http://localhost:5005/plynx/api/v0/graphs/5d292406713b286094d4ad87/
↪nodes/create_link' \
  -u default: -H "Content-Type: application/json" \
  -d '{"from": {"node_id": "5d2fbdf3373d3b7ce6e69043", "resource": "out"}, "to": {
↪"node_id": "5d3081ea99d54c7b6b8ff56b", "resource": "input"}}'

curl -X POST 'http://localhost:5005/plynx/api/v0/graphs/5d292406713b286094d4ad87/
↪nodes/change_parameter' \
  -u default: -H "Content-Type: application/json" \
  -d '{"node_id": "5d30b7eb88fb6a42caf0c565", "parameter_name": "template",
↪"parameter_value": "abc"}'
```

1.6.11 Working with a single Resource

This endpoint is a proxy between the client and internal Plynx resources.

WARNING: try to avoid calling this endpoint without “*preview*” argument set to *True*. Currently Plynx supports multiple data storages and is not optimized for a particular one. It will be fixed in the future versions, exposing additional endpoints.

Endpoint	HTTP Method
plynx/api/v0/resources/{resource_id}	GET

Blank line required after table.

Additional arguments to the endpoint:

Argument	Type	Description
preview	BOOLEAN	Preview flag (default: <i>false</i>)
file_type	STRING	One of the plugins. See <code>plynx-internal-file-types</code> for more details.

1.6.12 Get state of the Master

When Master is running, it periodically syncs its state with Plynx database. Use this endpoint to access it. See `plynx-internal-master_state`.

Endpoint	HTTP Method
plynx/api/v0/master_state	GET

1.6.12.1 Example

```
curl 'http://localhost:5005/plynx/api/v0/master_state' -u default:
```

Warning: Internal data structures are outdated. Please contact us for details.

1.7 Internal data structures

This section is meant for advanced users. Plynx exposes Rest API for creating Graphs and Nodes and monitoring their states. We will give you an overview of these objects below.

Table of Contents

- [Graph](#)
- [Node](#)

1.7.1 Graph

Graph is a basic structure of experiments. The Graph defines layout of Operations and Files and their dependencies. See [intro-graph](#).

Field Name	Type	Description
<code>_id</code>	STRING	Unique ID of the Graph.
<code>title</code>	STRING	Title string of the Graph.
<code>description</code>	STRING	Description string of the Graph.
<code>author</code>	STRING	ID of the author.
<code>graph_running_status</code>	STRING	Current status. Graph Running Status
<code>insertion_date</code>	STRING	Date time when the record was created in format <code>%Y-%m-%d %H:%M:%S.%LZ</code> .
<code>update_date</code>	STRING	Date time when the record was updated in format <code>%Y-%m-%d %H:%M:%S.%LZ</code> .
<code>nodes</code>	LIST	List for the nodes. See Node .
<code>_readonly</code>	BOOLEAN	Flag that shows if current user has write access to this object.

1.7.1.1 Graph Running Status

Value	Description
CREATED	Initial state of the Graph. Users can make changes in it.
READY	The Graph was approved. No changes accepted.
RUNNING	Master took control over the Graph and is working on execution.
SUCCESS	Execution of the Graph finished successfully.
FAILED_WAITING	One or more Operations failed during execution. Waiting for the rest of operations to be canceled.
FAILED	One or more Operations failed during execution.
CANCELED	User canceled execution.

1.7.2 Node

Operations and Files are derived from Nodes. See [intro-node](#).

Field Name	Type	Description
<code>_id</code>	STRING	Unique ID of the Node.
<code>title</code>	STRING	Title string of the Node.
<code>description</code>	STRING	Description string of the Node.
<code>author</code>	STRING	ID of the author.
<code>node_running_status</code>	STRING	Status of the Node inside a Graph. Node Running Status
<code>node_status</code>	STRING	Status of the Node. Node Status
<code>insertion_date</code>	STRING	Date time when the record was created in format <code>%Y-%m-%d %H:%M:%S.%LZ</code> .
<code>update_date</code>	STRING	Date time when the record was updated in format <code>%Y-%m-%d %H:%M:%S.%LZ</code> .
<code>_readonly</code>	BOOLEAN	Flag that shows if current user has write access to this object.
<code>base_node_name</code>	STRING	Base Node name. <code>plynx-concepts-base_node_name</code>
<code>cache_url</code>	STRING	URL to the cached operation.
<code>parent_node_id</code>	STRING	Reference to the parent Node. It refers to <code>nodes</code> collection.
<code>starred</code>	BOOLEAN	Flag is set to <code>true</code> if the Node is prioritized.
<code>successor_node_id</code>	STRING	If the Node is deprecated this field will reference to a successor in <code>nodes</code> collection.
<code>x</code>	INTEGER	<i>X</i> position in the Graph.
<code>y</code>	INTEGER	<i>Y</i> position in the Graph.
<code>inputs</code>	LIST	List for the Inputs. See Input .
<code>parameters</code>	LIST	List for the Parameters. See Parameter .
<code>logs</code>	LIST	List for the Logs. See Log .
<code>outputs</code>	LIST	List for the Outputs. See Output .

1.7.2.1 Input

Field Name	Type	Description
<code>name</code>	STRING	Name of the Input.
<code>file_types</code>	LIST	List of file types. See <code>plynx-plugins-file_types</code> .
<code>values</code>	LIST	List of Values. See Input Value .
<code>min_count</code>	INTEGER	Minimum number of Inputs.
<code>max_count</code>	INTEGER	Maximum number of Inputs.

1.7.2.2 Input Value

Field Name	Type	Description
node_id	STRING	ID of the Node the Operation depends on in a Graph.
output_id	STRING	Name of the Output in the Operation it depends on.
resource_id	STRING	Reference to a resource.

1.7.2.3 Parameter

Field Name	Type	Description
name	STRING	Name of the Parameter.
parameter_type	LIST	List of parameter types. See intro-parameter_types.
value	–	Type is specific to parameter type.
mutable_type	BOOLEAN	Flag specifies if user can change <i>parameter_type</i> .
removable	BOOLEAN	Flag specifies if user can remove the Parameter.
publicable	BOOLEAN	Flag specifies if user can publish the Parameter.
widget	OBJECT	<i>Null</i> of <i>Object</i> with the field <i>alias</i> . It contains the name of the parameter in the UI.

1.7.2.4 Output

Field Name	Type	Description
name	STRING	Name of the Output.
file_type	STRING	File type. See plynx-plugins-file_types.
resource_id	STRING	Reference to the file.

1.7.2.5 Log

Similar to *Output*. Field `file_type` is always set to `file`.

1.7.2.6 Node Running Status

This enum describes the state of the Node in the Graph.

Value	Description
STATIC	Usually Files have this status. This status never change.
CREATED	Initial state of the Node in the Graph.
IN_QUEUE	Operation is ready to be executed.
RUNNING	A worker is working on execution.
SUCCESS	Operation has been completed
RESTORED	Result of the Operation has been restored from previous execution.
FAILED	Operation has failed during execution.
CANCELED	User canceled execution.

1.7.2.7 Node Status

This enum describes global the state of the Node.

Value	Description
CREATED	Initial state of the Node. Users can still Modify it.
READY	Operation is ready to be used in Graphs.
DEPRECATED	Usage of the Operation is not recommended but it can be used.
MANDATORY_DEPRECATED	Usage of the Operation is not allowed.

1.8 Kubernetes deployment

This section covers basic deployment to google cloud kubernetes service.

Table of Contents

- *Step 0: Create a cluster*
- *Step 1: Create service account credentials*
- *Step 2: Create secret key*
- *Step 3: Create MongoDB pod*
- *Step 4: Plynx pods*
- *Step 5: Init users*
- *Step 6: Try Plynx*

1.8.1 Step 0: Create a cluster

Kubernetes on Google Cloud Quickstart

First we will need to set up *gcloud* utils.

```
gcloud config set project [YOUR_PROJECT_ID]
gcloud config set compute/zone [YOUR_ZONE]
```

Create cluster with a custom name.

```
export CLUSTER_NAME=[CLUSTER_NAME]
gcloud container clusters create ${CLUSTER_NAME}
gcloud container clusters get-credentials ${CLUSTER_NAME}
```

Resize cluster (optional).

```
# Optional
gcloud container clusters resize ${CLUSTER_NAME} --node-pool default-pool --num-nodes_
↪ 3
```

1.8.2 Step 1: Create service account credentials

Reference to Google Cloud docs

1. Create service account: *IAM & admin -> Service accounts -> Create Service Account*
2. Select custom username.
3. Add roles: *Storage Object Creator* and *Storage Object Viewer*
4. Create *json* key. It will download a file *plynx-*.json*, i.e. */Users/username/Downloads/plynx-197007-2aeb7faedf34.json*
5. Create a new bucket

```
# example YOUR_BUCKET_PATH=gs://plynx-test
export BUCKET=[YOUR_BUCKET_PATH]
gsutil mb ${BUCKET}
```

6. **IMPORTANT!** Add legacy permissions in Console User Interface.
 - a. Go to *Storage -> [YOUR_BUCKET_PATH] -> Permissions*.
 - b. In your User Account modify *Role(s)*: add *Storage Legacy Bucket Reader* and *Storage Legacy Bucket Writer*
7. Store credentials in kubernetes.

```
kubectl create secret generic gs-key --from-file=key.json=[PATH_TO_KEY_JSON]
```

8. Configure env variable mapping

```
kubectl create configmap storage-config --from-literal=storage-scheme=gs --from-
↪literal=storage-prefix=${BUCKET}/resources/
```

1.8.3 Step 2: Create secret key

Generate new secret key and write it to the file. Reuse the file in kubernetes secrets.

```
openssl rand -base64 16 | tr -d '\n' > secret.txt
kubectl create secret generic secret-key --from-file=secret.txt=./secret.txt
```

1.8.4 Step 3: Create MongoDB pod

Clone configuration files.

```
git clone https://github.com/plynx-team/plynx.git
cd plynx/kubernetes
```

To create the MongoDB pod, run these two commands:

```
kubectl apply -f googlecloud_ssd.yaml
kubectl apply -f mongo-statefulset.yaml
```

1.8.5 Step 4: Plynx pods

Create Plynx pods and services.

```
kubectl apply -f backend-service.yaml
kubectl apply -f backend-deployment.yaml
kubectl expose deployment backend --type=NodePort --name=backend-server

kubectl apply -f frontend-deployment.yaml
kubectl apply -f frontend-service.yaml

kubectl apply -f router.yaml

kubectl apply -f master-service.yaml
kubectl apply -f master-deployment.yaml

kubectl apply -f workers-deployment.yaml
```

1.8.6 Step 5: Init users

List of pods:

```
kubectl get pods

# NAME                                READY   STATUS    RESTARTS   AGE
# backend-8665dc7967-7wlks            1/1     Running   0           9m49s
# frontend-57857fc888-6gj57          1/1     Running   0           124m
# master-7f686d64f6-6shbq            1/1     Running   0           122m
# mongo-0                             2/2     Running   0           144m
# worker-6d5fc66f55-5g7q2            1/1     Running   5           76m
# worker-6d5fc66f55-5tsdf            1/1     Running   0           11m
# worker-6d5fc66f55-9vjv8            1/1     Running   0           11m
```

ssh to master pod.

```
kubectl exec -t -i master-7f686d64f6-6shbq bash
```

When connected, create a user.

```
plynx users --mode create_user --db-host mongo --username foo --password woo
```

1.8.7 Step 6: Try Plynx

1. Go to *Kubernetes Engine -> Services and Ingress*
2. Select Ingress called *api-router*
3. Go to the page located at *Load balancer IP*.
4. Use username *foo* and password *woo*

1.9 Frequently Asked Questions

1.9.1 Who is it for?

PLynx has been developed mainly for Data Scientists and ML Engineers as a high level abstraction of data pipelines and infrastructure. It is not limited by them and can be used by other technical and non-technical professionals, etc.

Modular architecture makes PLYnx rather a graph editor that transforms graphs into executable code.

1.9.2 Can I run it without docker?

Yes you can, but this way is not recommended.

Docker is an extremely powerful platform that has become a de-facto standard in the industry. Besides encapsulation, isolation and portability, it provides a more convenient way to control complex systems.

If you still want to run it without docker or have some other issues with deployment, please don't hesitate to contact us in [discord](#)

1.9.3 How is it different from Airflow, Kubeflow and other platforms?

Here are some main differences and principles.

- PLYnx follows the principle of rapid development: Try Fast Fail Fast. Highly reliable execution is not as important as flexibility and ability to try different experiments. Using reliable data pipelines in Data Science can bring incremental improvements, however there is usually far more to gain from other activities like integrating new data sources or using additional workflows.
- Each Operation has named *inputs* and *outputs*. It removes hidden logic and allows you to *reuse* existing Operations in new Workflows.
- The interface abstracts data scientists from the engineering and organizational complexities, such as data access, containerization, distributed processing, automatic failover, and other advanced computer science concepts.
- Plugins are very flexible and support multiple platforms and use cases.
- It encourages people to create their workflows in a more modular and parameterized way reusing existing solutions. This way they don't reinvent existing solutions many times and can use advantages of cached results and distributed computation.
- No need to have a domain specialist run the entire pipeline. Non-specialist can rerun an existing one.
- Experiments in Graph format are very well interpretable. It can be used by non-experts, or by people from other teams.

1.9.4 Is it a no-code platform?

Not exactly. Users have ability to write their own Operations as well as use existing repositories.

1.9.5 How can I install additional packages?

Option 1. Build your new images (preferred)

- Create a new Dockerfile with your dependancies.
- Install [PyPI PLYnx package](#)

- Deploy your new image.

Option 2. Run worker locally. (experimental)

- Run `make up_local_service` to start the database, UI and api services in docker-compose.
- Run `make up_local_worker` to start a single local worker. Note you will need to install packages necessary to run a worker itself.

Option 3. Local build copy.

- Add dependancies to your `requirements.txt`
- Run `make build` to build new images.
- Run `make up` to start your local Plynx.

Option 4. Kubernetes Operation (not available now)

Currently work in queue. [Please upvote](#)

1.9.6 How to contact us?

Please don't hesitate to join us in [discord](#).

1.10 License



Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms **and** conditions **for** use, reproduction, **and** distribution **as** defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner **or** entity authorized by the copyright owner that **is** granting the License.

"Legal Entity" shall mean the union of the acting entity **and** all other entities that control, are controlled by, **or** are under common control **with** that entity. For the purposes of this definition, "control" means (i) the power, direct **or** indirect, to cause the direction **or** management of such entity, whether by contract **or** otherwise, **or** (ii) ownership of fifty percent (50%) **or** more of the outstanding shares, **or** (iii) beneficial ownership of such entity.

"You" (**or** "Your") shall mean an individual **or** Legal Entity

(continues on next page)

(continued from previous page)

exercising permissions granted by this License.

"Source" form shall mean the preferred form **for** making modifications, including but **not** limited to software source code, documentation source, **and** configuration files.

"Object" form shall mean **any** form resulting **from mechanical** transformation **or** translation of a Source form, including but **not** limited to compiled **object** code, generated documentation, **and** conversions to other media types.

"Work" shall mean the work of authorship, whether **in** Source **or** Object form, made available under the License, **as** indicated by a copyright notice that **is** included **in or** attached to the work (an example **is** provided **in** the Appendix below).

"Derivative Works" shall mean **any** work, whether **in** Source **or** Object form, that **is** based on (**or** derived from) the Work **and for** which the editorial revisions, annotations, elaborations, **or** other modifications represent, **as** a whole, an original work of authorship. For the purposes of this License, Derivative Works shall **not** include works that remain separable from, **or** merely link (**or** bind by name) to the interfaces of, the Work **and** Derivative Works thereof.

"Contribution" shall mean **any** work of authorship, including the original version of the Work **and any** modifications **or** additions to that Work **or** Derivative Works thereof, that **is** intentionally submitted to Licensor **for** inclusion **in** the Work by the copyright owner **or** by an individual **or** Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means **any** form of electronic, verbal, **or** written communication sent to the Licensor **or** its representatives, including but **not** limited to communication on electronic mailing lists, source code control systems, **and** issue tracking systems that are managed by, **or** on behalf of, the Licensor **for** the purpose of discussing **and** improving the Work, but excluding communication that **is** conspicuously marked **or** otherwise designated **in** writing by the copyright owner **as** "Not a Contribution."

"Contributor" shall mean Licensor **and any** individual **or** Legal Entity on behalf of whom a Contribution has been received by Licensor **and** subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms **and** conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, **and** distribute the Work **and** such Derivative Works **in** Source **or** Object form.
3. Grant of Patent License. Subject to the terms **and** conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (**except as** stated **in** this section) patent license to make, have made, use, offer to sell, sell, import, **and** otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone **or** by combination of their Contribution(s)

(continues on next page)

(continued from previous page)

with the Work to which such Contribution(s) was submitted. If You institute patent litigation against **any** entity (including a cross-claim **or** counterclaim **in** a lawsuit) alleging that the Work **or** a Contribution incorporated within the Work constitutes direct **or** contributory patent infringement, then **any** patent licenses granted to You under this License **for** that Work shall terminate **as** of the date such litigation **is** filed.

4. Redistribution. You may reproduce **and** distribute copies of the Work **or** Derivative Works thereof **in any** medium, **with or** without modifications, **and in** Source **or** Object form, provided that You meet the following conditions:
 - (a) You must give **any** other recipients of the Work **or** Derivative Works a copy of this License; **and**
 - (b) You must cause **any** modified files to carry prominent notices stating that You changed the files; **and**
 - (c) You must retain, **in** the Source form of **any** Derivative Works that You distribute, **all** copyright, patent, trademark, **and** attribution notices **from the** Source form of the Work, excluding those notices that do **not** pertain to **any** part of the Derivative Works; **and**
 - (d) If the Work includes a "NOTICE" text file **as** part of its distribution, then **any** Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do **not** pertain to **any** part of the Derivative Works, **in** at least one of the following places: within a NOTICE text file distributed **as** part of the Derivative Works; within the Source form **or** documentation, **if** provided along **with** the Derivative Works; **or**, within a display generated by the Derivative Works, **if and** wherever such third-party notices normally appear. The contents of the NOTICE file are **for** informational purposes only **and** do **not** modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside **or as** an addendum to the NOTICE text **from the** Work, provided that such additional attribution notices cannot be construed **as** modifying the License.

You may add Your own copyright statement to Your modifications **and** may provide additional **or** different license terms **and** conditions **for** use, reproduction, **or** distribution of Your modifications, **or for any** such Derivative Works **as** a whole, provided Your use, reproduction, **and** distribution of the Work otherwise complies **with** the conditions stated **in** this License.

5. Submission of Contributions. Unless You explicitly state otherwise, **any** Contribution intentionally submitted **for** inclusion **in** the Work by You to the Licensors shall be under the terms **and** conditions of this License, without **any** additional terms **or** conditions. Notwithstanding the above, nothing herein shall supersede **or** modify the terms of **any** separate license agreement you may have executed **with** Licensors regarding such Contributions.

(continues on next page)

(continued from previous page)

6. Trademarks. This License does **not** grant permission to use the trade names, trademarks, service marks, **or** product names of the Licensor, **except as** required **for** reasonable **and** customary use **in** describing the origin of the Work **and** reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law **or** agreed to **in** writing, Licensor provides the Work (**and** each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express **or** implied, including, without limitation, **any** warranties **or** conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, **or** FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible **for** determining the appropriateness of using **or** redistributing the Work **and** assume **any** risks associated **with** Your exercise of permissions under this License.
8. Limitation of Liability. In no event **and** under no legal theory, whether **in** tort (including negligence), contract, **or** otherwise, unless required by applicable law (such **as** deliberate **and** grossly negligent acts) **or** agreed to **in** writing, shall **any** Contributor be liable to You **for** damages, including **any** direct, indirect, special, incidental, **or** consequential damages of **any** character arising **as** a result of this License **or** out of the use **or** inability to use the Work (including but **not** limited to damages **for** loss of goodwill, work stoppage, computer failure **or** malfunction, **or** **any** **and** all other commercial damages **or** losses), even **if** such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty **or** Additional Liability. While redistributing the Work **or** Derivative Works thereof, You may choose to offer, **and** charge a fee **for**, acceptance of support, warranty, indemnity, **or** other liability obligations **and/or** rights consistent **with** this License. However, **in** accepting such obligations, You may act only on Your own behalf **and** on Your sole responsibility, **not** on behalf of **any** other Contributor, **and** only **if** You agree to indemnify, defend, **and** hold each Contributor harmless **for** **any** liability incurred by, **or** claims asserted against, such Contributor by reason of your accepting **any** such warranty **or** additional liability.

1.11 API Reference

This page contains auto-generated API reference documentation¹.

1.11.1 plynx

Interactive, Scalable, Shareable and Reproducible Workflow Orchestration framework

1.11.1.1 Subpackages

`plynx.base`

¹ Created with `sphinx-autoapi`

Submodules

`plynx.base.executor`

Templates for Plynx Executors and utils.

Module Contents

class `plynx.base.executor.RunningStatus`

Async job running status

node_running_status :str

class `plynx.base.executor.BaseExecutor` (*node: Optional[Node] = None*)

Bases: `abc.ABC`

Base Executor class

IS_GRAPH :bool = False

_update_node (*self, node*)

run (*self, preview: bool = False*)

Main execution function.

- Workdir has been initialized.
- Inputs are not preprocessed.
- Outputs should be manually postprocessed.
- It is OK to raise an exception in this function.

Returns: enum: `plynx.constants.NodeRunningStatus`

launch (*self*)

Launch the Node on the backend.

The difference between *launch()* and *run()* is that: - *run()* assumes synchronous execution. - *launch()* does not necessarily have this assumption. Use *get_node_running_status* to get the status of the execution.

Returns: `RunningStatus`

get_running_status (*self*)

Returns the status of the execution.

Async executions should sync with the remote and return the result immediately.

kill (*self*)

Force to kill the process.

The reason can be the fact it was working too long or parent executor canceled it.

is_updated (*self*)

Function that is regularly called by a Worker.

The function is running in a separate thread and does not block execution of *run()*.

Returns: (bool): True if worker needs to update DB else False

classmethod **get_default_node** (*cls, is_workflow: bool*)

Generate a new default Node for this executor

```
init_executor (self)
    Initialize environment for the executor

clean_up_executor (self)
    Clean up the environment created by executor

validate (self, ignore_inputs: bool = True)
    Validate Node.

    Return: (ValidationError) Validation error if found; else None
```

```
class plynx.base.executor.Dummy
    Bases: plynx.base.executor.BaseExecutor

    Dummy Executor. Used for static Operations

    run (self, preview=False)
        Not Implemented

    status (self)
        Not Implemented

    kill (self)
        Not Implemented

    classmethod get_default_node (cls, is_workflow: bool)
        Not Implemented
```

plynx.base.hub

Templates for PLYnx Hubs and utils.

Module Contents

```
class plynx.base.hub.Query
    Hub search query entry

    status :str =
    search :str =
    per_page :int = 30
    offset :int = 0
    user_id :Optional[str]

class plynx.base.hub.BaseHub
    Base Hub class

    search (self, query: Query)
        Search for items given a query
```

plynx.base.resource

Templates for PLYnx Resources and utils.

Module Contents

```

plynx.base.resource.PreviewObject
plynx.base.resource.__force_decode (byte_array)
class plynx.base.resource.BaseResource
    Base Resource class

    DISPLAY_RAW :bool = False
    DISPLAY_THUMBNAIL :bool = False

    static prepare_input (filename: str, preview: bool = False)
        Resource preprocessor

    static prepare_output (filename: str, preview: bool = False)
        Prepare output

    static preprocess_input (value: Any)
        Resource preprocessor

    static postprocess_output (value: Any)
        Resource postprocessor

    classmethod preview (cls, preview_object: PreviewObject)
        Preview Resource

    classmethod thumbnail (cls, output: Any)
        Thumbnail preview Resource

```

plynx.bin

PLynx CLI.

Submodules

plynx.bin.cli

PLynx CLI parser

Module Contents

```

plynx.bin.cli._config
class plynx.bin.cli.Arg
    Common argument tuple

    flags :Union[Tuple[str], Tuple[str, str]]
    help :str
    action :Optional[str]
    default :Optional[Any]
    type :Optional[Type]
    levels :Optional[List[str]]

```

```

    required :bool = False
    nargs :Optional[str]

plynx.bin.cli.api (args)
    Start web service.

plynx.bin.cli.worker_server (args)
    Start worker service.

plynx.bin.cli.cache (args)
    Show cache options.

plynx.bin.cli.worker (args)
    Start worker service.

plynx.bin.cli.users (args)
    Show users options.

plynx.bin.cli.version (args)
    Print Plynx version

plynx.bin.cli.execute (args)
    Execute Operation.

plynx.bin.cli.make_operations_meta (args)
    Execute Operation.

class plynx.bin.cli.CLIFactory
    The class that generates Plynx CLI parser

    ARGS

    SUBPARSERS

    classmethod parse_global_config_parameters (cls, args)
        Parse parameters applied to all of the services.

    classmethod get_parser (cls)
        Generate CLI parser

plynx.bin.cli.get_parser ()
    Generate CLI parser

```

Package Contents

```

class plynx.bin.CLIFactory
    The class that generates Plynx CLI parser

    ARGS

    SUBPARSERS

    classmethod parse_global_config_parameters (cls, args)
        Parse parameters applied to all of the services.

    classmethod get_parser (cls)
        Generate CLI parser

plynx.bin.main ()
    Main Plynx CLI function.

```

`plynx.constants`

Main Plynx constants defined in this module

Submodules

`plynx.constants.collections`

Standard Plynx collections in DB

Module Contents

```
class plynx.constants.collections.Collections
    All basic collections used in DB

    NODE_CACHE :str = node_cache
    RUN_CANCELLATIONS :str = run_cancellations
    RUNS :str = runs
    TEMPLATES :str = templates
    USERS :str = users
    WORKER_STATE :str = worker_state
    HUB_NODE_REGISTRY :str = hub_node_registry
```

`plynx.constants.node_enums`

Node constants

Module Contents

```
class plynx.constants.node_enums.NodeRunningStatus
    Running status enum

    STATIC :str = STATIC
    CREATED :str = CREATED
    READY :str = READY
    IN_QUEUE :str = IN_QUEUE
    RUNNING :str = RUNNING
    SUCCESS :str = SUCCESS
    RESTORED :str = RESTORED
    FAILED :str = FAILED
    FAILED_WAITING :str = FAILED_WAITING
    CANCELED :str = CANCELED
```

```

    SPECIAL :str = SPECIAL
    _FAILED_STATUSES :Set[str]
    _SUCCEEDED_STATUSES :Set[str]
    _AWAITING_STATUSES :Set[str]
    _NON_CHANGEABLE_STATUSES :Set[str]
    _FINISHED_STATUSES :Set[str]
    static is_finished(node_running_status: str)
        Check if the status is final
    static is_succeeded(node_running_status: str)
        Check if the status is final and successful
    static is_failed(node_running_status: str)
        Check if the status is final and failed
    static is_non_changeable(node_running_status: str)
        Check if the status is in static or special
class plynx.constants.node_enums.NodeStatus
    Node permanent status
    CREATED :str = CREATED
    READY :str = READY
    DEPRECATED :str = DEPRECATED
    MANDATORY_DEPRECATED :str = MANDATORY_DEPRECATED
class plynx.constants.node_enums.NodePostAction
    HTTP post action
    SAVE :str = SAVE
    APPROVE :str = APPROVE
    CREATE_RUN :str = CREATE_RUN
    CREATE_RUN_FROM_SCRATCH :str = CREATE_RUN_FROM_SCRATCH
    CLONE :str = CLONE
    VALIDATE :str = VALIDATE
    DEPRECATE :str = DEPRECATE
    MANDATORY_DEPRECATE :str = MANDATORY_DEPRECATE
    PREVIEW_CMD :str = PREVIEW_CMD
    REARRANGE_NODES :str = REARRANGE_NODES
    UPGRADE_NODES :str = UPGRADE_NODES
    CANCEL :str = CANCEL
    GENERATE_CODE :str = GENERATE_CODE
class plynx.constants.node_enums.NodePostStatus
    Standard HTTP response status
    SUCCESS :str = SUCCESS

```

```
    FAILED :str = FAILED
    VALIDATION_FAILED :str = VALIDATION_FAILED
class plynx.constants.node_enums.NodeClonePolicy
    Clone algorithm
    NODE_TO_NODE :int = 0
    NODE_TO_RUN :int = 1
    RUN_TO_NODE :int = 2
class plynx.constants.node_enums.NodeVirtualCollection
    Virtual collection
    OPERATIONS :str = operations
    WORKFLOWS :str = workflows
class plynx.constants.node_enums.SpecialNodeId
    Special Node IDs in the workflows
    INPUT :ObjectId
    OUTPUT :ObjectId
class plynx.constants.node_enums.NodeOrigin
    Enum that indicates where the Node came from
    DB :str = DB
    BUILT_IN_HUBS :str = BUILT_IN_HUBS
plynx.constants.node_enums.IGNORED_CACHE_PARAMETERS
```

`plynx.constants.parameter_types`

Parameter enums

Module Contents

```
class plynx.constants.parameter_types.ParameterTypes
    Standard parameter types
    STR :str = str
    INT :str = int
    FLOAT :str = float
    BOOL :str = bool
    TEXT :str = text
    ENUM :str = enum
    LIST_STR :str = list_str
    LIST_INT :str = list_int
    LIST_NODE :str = list_node
    CODE :str = code
```

```
COLOR :str = color
```

```
plynx.constants.parameter_types.PRIMITIVE_TYPES
```

```
plynx.constants.resource_enums
```

Resource enums

Module Contents

```
class plynx.constants.resource_enums.NodeResources
```

Internal node elements

```
INPUT :str = inputs
```

```
OUTPUT :str = outputs
```

```
CLOUD_INPUT :str = cloud_inputs
```

```
CLOUD_OUTPUT :str = cloud_outputs
```

```
PARAM :str = params
```

```
LOG :str = logs
```

```
class plynx.constants.resource_enums.HubSearchParams
```

Describing search based on resource

```
INPUT_FILE_TYPE :str = input_file_type
```

```
OUTPUT_FILE_TYPE :str = output_file_type
```

```
plynx.constants.users
```

User based enums

Module Contents

```
class plynx.constants.users.IAMPolicies
```

Standard role policies

```
CAN_VIEW_OTHERS_OPERATIONS :str = CAN_VIEW_OTHERS_OPERATIONS
```

```
CAN_VIEW_OTHERS_WORKFLOWS :str = CAN_VIEW_OTHERS_WORKFLOWS
```

```
CAN_VIEW_OPERATIONS :str = CAN_VIEW_OPERATIONS
```

```
CAN_VIEW_WORKFLOWS :str = CAN_VIEW_WORKFLOWS
```

```
CAN_CREATE_OPERATIONS :str = CAN_CREATE_OPERATIONS
```

```
CAN_CREATE_WORKFLOWS :str = CAN_CREATE_WORKFLOWS
```

```
CAN_MODIFY_OTHERS_WORKFLOWS :str = CAN_MODIFY_OTHERS_WORKFLOWS
```

```
CAN_RUN_WORKFLOWS :str = CAN_RUN_WORKFLOWS
```

```
IS_ADMIN :str = IS_ADMIN
```

```
class plynx.constants.users.UserPostAction
    HTTP POST action options

    MODIFY :str = MODIFY

class plynx.constants.users.RegisterUserExceptionCode
    Validation error codes

    EMPTY_USERNAME :str = EMPTY_USERNAME

    EMPTY_PASSWORD :str = EMPTY_PASSWORD

    USERNAME_ALREADY_EXISTS :str = USERNAME_ALREADY_EXISTS

    EMAIL_ALREADY_EXISTS :str = EMAIL_ALREADY_EXISTS

    INVALID_EMAIL :str = INVALID_EMAIL

    INVALID_LENGTH_OF_USERNAME :str = INVALID_LENGTH_OF_USERNAME

class plynx.constants.users.TokenType
    Auth token type

    ACCESS_TOKEN = access

    REFRESH_TOKEN = refresh
```

`plynx.constants.validation_enums`

Node validation enums

Module Contents

```
class plynx.constants.validation_enums.ValidationTargetType
    Object Target

    BLOCK :str = BLOCK

    GRAPH :str = GRAPH

    INPUT :str = INPUT

    NODE :str = NODE

    PARAMETER :str = PARAMETER

    PROPERTY :str = PROPERTY

class plynx.constants.validation_enums.ValidationCode
    Standard validation code

    IN_DEPENDENTS :str = IN_DEPENDENTS

    MISSING_INPUT :str = MISSING_INPUT

    MISSING_PARAMETER :str = MISSING_PARAMETER

    INVALID_VALUE :str = INVALID_VALUE

    DEPRECATED_NODE :str = DEPRECATED_NODE

    EMPTY_GRAPH :str = EMPTY_GRAPH
```

`plynx.constants.web`

Web constants

Module Contents

```
class plynx.constants.web.ResponseStatus
    Returned response status

    SUCCESS :str = SUCCESS

    FAILED :str = FAILED
```

Package Contents

```
class plynx.constants.Collections
    All basic collections used in DB

    NODE_CACHE :str = node_cache

    RUN_CANCELLATIONS :str = run_cancellations

    RUNS :str = runs

    TEMPLATES :str = templates

    USERS :str = users

    WORKER_STATE :str = worker_state

    HUB_NODE_REGISTRY :str = hub_node_registry
```

`plynx.constants.IGNORED_CACHE_PARAMETERS`

```
class plynx.constants.NodeClonePolicy
    Clone algorithm

    NODE_TO_NODE :int = 0

    NODE_TO_RUN :int = 1

    RUN_TO_NODE :int = 2
```

```
class plynx.constants.NodeOrigin
    Enum that indicates where the Node came from

    DB :str = DB

    BUILT_IN_HUBS :str = BUILT_IN_HUBS
```

```
class plynx.constants.NodePostAction
    HTTP post action

    SAVE :str = SAVE

    APPROVE :str = APPROVE

    CREATE_RUN :str = CREATE_RUN

    CREATE_RUN_FROM_SCRATCH :str = CREATE_RUN_FROM_SCRATCH

    CLONE :str = CLONE

    VALIDATE :str = VALIDATE
```

```
DEPRECATE :str = DEPRECATE
MANDATORY_DEPRECATE :str = MANDATORY_DEPRECATE
PREVIEW_CMD :str = PREVIEW_CMD
REARRANGE_NODES :str = REARRANGE_NODES
UPGRADE_NODES :str = UPGRADE_NODES
CANCEL :str = CANCEL
GENERATE_CODE :str = GENERATE_CODE

class plynx.constants.NodePostStatus
    Standard HTTP response status

    SUCCESS :str = SUCCESS
    FAILED :str = FAILED
    VALIDATION_FAILED :str = VALIDATION_FAILED

class plynx.constants.NodeRunningStatus
    Running status enum

    STATIC :str = STATIC
    CREATED :str = CREATED
    READY :str = READY
    IN_QUEUE :str = IN_QUEUE
    RUNNING :str = RUNNING
    SUCCESS :str = SUCCESS
    RESTORED :str = RESTORED
    FAILED :str = FAILED
    FAILED_WAITING :str = FAILED_WAITING
    CANCELED :str = CANCELED
    SPECIAL :str = SPECIAL
    _FAILED_STATUSES :Set[str]
    _SUCCEEDED_STATUSES :Set[str]
    _AWAITING_STATUSES :Set[str]
    _NON_CHANGEABLE_STATUSES :Set[str]
    _FINISHED_STATUSES :Set[str]

    static is_finished (node_running_status: str)
        Check if the status is final

    static is_succeeded (node_running_status: str)
        Check if the status is final and successful

    static is_failed (node_running_status: str)
        Check if the status is final and failed

    static is_non_changeable (node_running_status: str)
        Check if the status is in static or special
```

```
class plynx.constants.NodeStatus
    Node permanent status

    CREATED :str = CREATED

    READY :str = READY

    DEPRECATED :str = DEPRECATED

    MANDATORY_DEPRECATED :str = MANDATORY_DEPRECATED

class plynx.constants.NodeVirtualCollection
    Virtual collection

    OPERATIONS :str = operations

    WORKFLOWS :str = workflows

class plynx.constants.SpecialNodeId
    Special Node IDs in the workflows

    INPUT :ObjectId

    OUTPUT :ObjectId

plynx.constants.PRIMITIVE_TYPES

class plynx.constants.ParameterTypes
    Standard parameter types

    STR :str = str

    INT :str = int

    FLOAT :str = float

    BOOL :str = bool

    TEXT :str = text

    ENUM :str = enum

    LIST_STR :str = list_str

    LIST_INT :str = list_int

    LIST_NODE :str = list_node

    CODE :str = code

    COLOR :str = color

class plynx.constants.HubSearchParams
    Describing search based on resource

    INPUT_FILE_TYPE :str = input_file_type

    OUTPUT_FILE_TYPE :str = output_file_type

class plynx.constants.NodeResources
    Internal node elements

    INPUT :str = inputs

    OUTPUT :str = outputs

    CLOUD_INPUT :str = cloud_inputs

    CLOUD_OUTPUT :str = cloud_outputs
```

```
PARAM :str = params
LOG :str = logs

class plynx.constants.IAMPolicies
    Standard role policies

    CAN_VIEW_OTHERS_OPERATIONS :str = CAN_VIEW_OTHERS_OPERATIONS
    CAN_VIEW_OTHERS_WORKFLOWS :str = CAN_VIEW_OTHERS_WORKFLOWS
    CAN_VIEW_OPERATIONS :str = CAN_VIEW_OPERATIONS
    CAN_VIEW_WORKFLOWS :str = CAN_VIEW_WORKFLOWS
    CAN_CREATE_OPERATIONS :str = CAN_CREATE_OPERATIONS
    CAN_CREATE_WORKFLOWS :str = CAN_CREATE_WORKFLOWS
    CAN_MODIFY_OTHERS_WORKFLOWS :str = CAN_MODIFY_OTHERS_WORKFLOWS
    CAN_RUN_WORKFLOWS :str = CAN_RUN_WORKFLOWS
    IS_ADMIN :str = IS_ADMIN

class plynx.constants.RegisterUserExceptionCode
    Validation error codes

    EMPTY_USERNAME :str = EMPTY_USERNAME
    EMPTY_PASSWORD :str = EMPTY_PASSWORD
    USERNAME_ALREADY_EXISTS :str = USERNAME_ALREADY_EXISTS
    EMAIL_ALREADY_EXISTS :str = EMAIL_ALREADY_EXISTS
    INVALID_EMAIL :str = INVALID_EMAIL
    INVALID_LENGTH_OF_USERNAME :str = INVALID_LENGTH_OF_USERNAME

class plynx.constants.TokenType
    Auth token type

    ACCESS_TOKEN = access
    REFRESH_TOKEN = refresh

class plynx.constants.UserPostAction
    HTTP POST action options

    MODIFY :str = MODIFY

class plynx.constants.ValidationCode
    Standard validation code

    IN_DEPENDENTS :str = IN_DEPENDENTS
    MISSING_INPUT :str = MISSING_INPUT
    MISSING_PARAMETER :str = MISSING_PARAMETER
    INVALID_VALUE :str = INVALID_VALUE
    DEPRECATED_NODE :str = DEPRECATED_NODE
    EMPTY_GRAPH :str = EMPTY_GRAPH

class plynx.constants.ValidationTargetType
    Object Target
```

```

BLOCK :str = BLOCK
GRAPH :str = GRAPH
INPUT :str = INPUT
NODE :str = NODE
PARAMETER :str = PARAMETER
PROPERTY :str = PROPERTY
class plynx.constants.ResponseStatus
    Returned response status
    SUCCESS :str = SUCCESS
    FAILED :str = FAILED

```

`plynx.db`

Submodules

`plynx.db.db_object`

The class defines *DBObject*. This is an abstraction of all of the objects in database.

Module Contents

```

plynx.db.db_object.DBObjectType
plynx.db.db_object._registry
plynx.db.db_object.register_class(target_class)
    Register inherited from DB Object class
plynx.db.db_object.get_class(name: str) → DBObjectType
    Get DB Object inherited class object by its name from the registry
exception plynx.db.db_object.DBObjectNotFound
    Bases: Exception
    Internal Exception.
exception plynx.db.db_object.ClassNotSavable
    Bases: Exception
    Internal Exception.
class plynx.db.db_object._DBObject
    DB Object. Abstraction of an object in the DB.
    DB_COLLECTION =
    classmethod load(cls: Type[DBObjectType], _id: ObjectId, collection: Optional[str] = None)
        Load object from db.
        Args: _id (ObjectId): ID of the object in DB
    save(self, force: bool = False, collection: Optional[str] = None)
        Save Object in the database

```

copy (*self: DBObjectType*)

Make a copy

Return: A copy of the Object

classmethod from_dict (*cls: Type[DBObjectType], dict_obj: Dict[str, Any]*)

Create a class based on dict_obj

to_dict (*self: DBObjectType*)

Create serialized object

__str__ (*self*)

__repr__ (*self*)

class `plynx.db.db_object.Meta`

Bases: `type`

Class Registry handle

classmethod **__new__** (*meta, name, bases, class_dict*)

class `plynx.db.db_object.DBObject`

Bases: `plynx.db.db_object._DBObject`

DB Object. Abstraction of an object in the DB.

Args: obj_dict (dict, None): Representation of the object. If None, an object with default fields will be created.

__post_init__ (*self*)

`plynx.db.demo_user_manager`

User Manager for demo

Module Contents

`plynx.db.demo_user_manager.template_collection_manager`

class `plynx.db.demo_user_manager.DemoUserManager`

The class contains Demo user code.

demo_config :`DemoConfig`

static **_id_generator** (*size: int = 6, chars: str = string.ascii_uppercase + string.digits*)

static **create_demo_user** ()

Create a demo user

static **create_demo_templates** (*user*)

Clone the default templates and assign them to the user.

`plynx.db.node`

Node DB Object and utils

Module Contents

```
plynx.db.node._clone_update_in_place (node: 'Node', node_clone_policy: int, override_finished_state: bool, override_node_id: bool = False)
```

```
class plynx.db.node._BaseResource
    Bases: plynx.db.db_object.DBObject

    name :str =
    file_type :str
    values :List[Any]
    is_array :bool = False
    min_count :int = 1
```

```
class plynx.db.node.Output
    Bases: plynx.db.node._BaseResource

    Basic Output structure.

    thumbnail :Optional[str]
```

```
class plynx.db.node.InputReference
    Bases: plynx.db.db_object.DBObject

    Basic Value of the Input structure.

    node_id :ObjectId
    output_id :str =
```

```
class plynx.db.node.Input
    Bases: plynx.db.node._BaseResource

    Basic Input structure.

    primitive_override :Any
    input_references :List[InputReference]

    __post_init__(self)
        Set default primitive_override if it is not set and file_type is primitive

    add_input_reference (self, node_id: ObjectId, output_id: str)
        Add input reference to the input
```

```
class plynx.db.node.CachedNode
    Bases: plynx.db.db_object.DBObject

    Values to override Node on display

    node_running_status :str
    outputs :List[Output]
    logs :List[Output]
```

```
class plynx.db.node.Node
    Bases: plynx.db.db_object.DBObject

    Basic Node with db interface.

    DB_COLLECTION
```

```
_id :ObjectId
_type :str = Node
_cached_node :Optional[CachedNode]
title :str = Title
description :str =
kind :str = dummy
parent_node_id :Optional[ObjectId]
successor_node_id :Optional[ObjectId]
original_node_id :Optional[ObjectId]
template_node_id :Optional[ObjectId]
origin :str
code_hash :str =
code_function_location :Optional[str]
node_running_status :str
node_status :str
cache_url :Optional[str]
x :int = 0
y :int = 0
author :Optional[ObjectId]
starred :bool = False
auto_sync :bool = True
auto_run :bool = True
auto_run_enabled :bool = True
latest_run_id :Optional[ObjectId]
inputs :List[Input]
parameters :List['Parameter']
outputs :List[Output]
logs :List[Output]
static _default_log (name: str)
apply_properties (self, other_node: 'Node')
    Apply Properties and Inputs of another Node. This method is used for updating nodes.
    Args: other_node (Node): A node to copy Properties and Inputs from
clone (self, node_clone_policy: int, override_finished_state: bool = True)
    Return a cloned copy of a Node
_get_custom_element (self, arr: Union[List[Input], List['Parameter'], List[Output]], name: str,
                        throw: bool, default: Optional[Callable[[str], Union[Input, 'Parameter',
                        Output]]] = None)
```

```

get_input_by_name (self, name: str)
    Find Input object

get_parameter_by_name (self, name: str)
    Find Parameter object

get_parameter_by_name_safe (self, name: str)
    Find Parameter object

get_output_by_name (self, name: str)
    Find Output object

get_log_by_name (self, name: str)
    Find Log object

get_sub_nodes (self)
    Get a list of subnodes

class plynx.db.node.ParameterEnum
    Bases: plynx.db.db_object.DBObject

    Enum value.

    values :List[str]

    index :int = 0

class plynx.db.node.ParameterCode
    Bases: plynx.db.db_object.DBObject

    Code value.

    value :str =

    mode :str = python

class plynx.db.node.ParameterListOfNodes
    Bases: plynx.db.db_object.DBObject

    List Of Nodes value.

    value :List[Node]

plynx.db.node._get_default_by_type (parameter_type: str)

plynx.db.node._value_is_valid (value, parameter_type: str)

class plynx.db.node.Parameter
    Bases: plynx.db.db_object.DBObject

    Basic Parameter structure.

    name :str =

    parameter_type :str

    value :Any =

    mutable_type :bool = True

    removable :bool = True

    publicable :bool = True

    widget :Optional[str]

    reference :Optional[str]

```

```
__post_init__(self)
```

`plynx.db.node_cache`

Node Cache and utils

Module Contents

```
plynx.db.node_cache.demo_config :DemoConfig
```

```
class plynx.db.node_cache.NodeCache
    Bases: plynx.db.db_object.DBObject
```

Basic Node Cache with db interface.

```
DB_COLLECTION
```

```
_id :ObjectId
```

```
key :str =
```

```
node_id :Optional[ObjectId]
```

```
run_id :Optional[ObjectId]
```

```
outputs :List[Output]
```

```
logs :List[Output]
```

```
removed :bool = False
```

```
protected :bool = False
```

```
static instantiate (node: Node, run_id: ObjectId)
```

Instantiate a Node Cache from Node.

Args: node (Node): Node object run_id (ObjectId): Run ID

Return: (NodeCache)

```
static generate_key (node: Node)
```

Generate hash.

Args: node (Node): Node object

Return: (str) Hash value

`plynx.db.node_cache_manager`

Cache Manager and utils.

Module Contents

```
class plynx.db.node_cache_manager.NodeCacheManager
    The Node cache interface.
```

The cache is defined by Node's

- original_node_id

- inputs
- parameters

static get (*node: Node*)
Pull NodeCache if exists.

Args: node (Node): Node object

Return: (NodeCache) NodeCache or None

static post (*node: Node, run_id: ObjectId*)
Create NodeCache instance in the database.

Args: node (Node): Node object run_id (ObjectId, str): Run ID

Return: True if cache saved else False

static _make_query (*start_datetime: Optional[datetime.datetime] = None, end_datetime: Optional[datetime.datetime] = None, non_protected_only: bool = False*)
Make sample query.

Args: start_datetime (datetime, None): Start datetime or None if selecting from beginning end_datetime (datetime, None): End datetime or None if selecting until now

Return: Iterator on the list of dict-like objects

static get_list (*start_datetime: Optional[datetime.datetime] = None, end_datetime: Optional[datetime.datetime] = None, non_protected_only: bool = False*)
List of NodeCache objects.

Args: start_datetime (datetime, None): Start datetime or None if selecting from beginning end_datetime (datetime, None): End datetime or None if selecting until now

Return: Iterator on the list of dict-like objects

static clean_up ()
Remove NodeCache objects with flag *removed* set

`plynx.db.node_collection_manager`

Node collection manager and utils

Module Contents

```
plynx.db.node_collection_manager._PROPERTIES_TO_GET_FROM_SUBS = ['node_running_status', 'l
```

class `plynx.db.node_collection_manager.NodeCollectionManager` (*collection: str*)
NodeCollectionManager contains all the operations to work with Nodes in the database.

get_db_objects (*self, status: Union[List[str], str] = "", node_kinds: Union[None, str, List[str]] = None, search: str = "", per_page: int = 20, offset: int = 0, user_id: Optional[ObjectId] = None*)
Get subset of the Objects.

Args: status (str, None): Node Running Status search (str, None): Search pattern per_page (int): Number of Nodes per page offset (int): Offset

Return: (list of dict) List of Nodes in dict format

get_db_objects_by_ids (*self*, *ids*: Union[List[ObjectId], List[str]], *collection*: Optional[str] = None)
Find all the Objects with a given IDs.
Args: *ids* (list of ObjectId): Object Ids

_update_sub_nodes_fields (*self*, *sub_nodes_dicts*: List[Dict], *reference_node_id*: str, *target_props*: List[str], *reference_collection*: Optional[str] = None)

get_db_node (*self*, *node_id*: ObjectId, *user_id*: Optional[ObjectId] = None)
Get dict representation of a Node.
Args: *node_id* (ObjectId, str): Object ID *user_id* (str, ObjectId, None): User ID
Return: (dict) dict representation of the Object

get_db_object (*self*, *object_id*: ObjectId, *user_id*: Optional[ObjectId] = None)
Get dict representation of an Object.
Args: *object_id* (ObjectId): Object ID *user_id* (ObjectId, None): User ID
Return: (dict) dict representation of the Object

static _transplant_node (*node*: Node, *dest_node*: Node)

upgrade_sub_nodes (*self*, *main_node*: Node)
Upgrade deprecated Nodes.
The function does not change the original graph in the database.
Return: (int): Number of upgraded Nodes

pick_node (*self*, *kinds*: List[str])
Get node and set status to RUNNING in atomic way

plynx.db.run_cancellation_manager

Cancellation DB Object and utils

Module Contents

class `plynx.db.run_cancellation_manager.RunCancellation`
Bases: `plynx.db.db_object.DBObject`
RunCancellation represents Run Cancellation event in the database.
DB_COLLECTION
_id :ObjectId
run_id :Optional[ObjectId]

class `plynx.db.run_cancellation_manager.RunCancellationManager`
RunCancellationManager contains basic operations related to *runs_cancellations* collection.
run_id :ObjectId
static cancel_run (*run_id*: ObjectId)
Cancel Run. Args:
Unexpected indentation.
run_id (ObjectId) RunID

```

static get_run_cancellations ()
    Get all Run Cancellation events

static remove (runs_cancellation_ids: List[ObjectId])
    Remove Run Cancellation events with given Ids Args:

    Unexpected indentation.

    runs_cancellation_ids (list of ObjectId) List of Run IDs to remove

```

plynx.db.user

User DB Object and utils

Module Contents

```

plynx.db.user.DEFAULT_POLICIES
plynx.db.user.JWT_ENCODE_ALGORITHM = HS256

class plynx.db.user.UserSettings
    Bases: plynx.db.db_object.DBObject

    User Settings structure.

    display_name :str =
    picture :str =

class plynx.db.user.User
    Bases: plynx.db.db_object.DBObject

    Basic User class with db interface.

    DB_COLLECTION

    _id :ObjectId
    username :str =
    email :Optional[str] =
    password_hash :str =
    active :bool = True
    policies :List[str]
    settings :UserSettings

    hash_password (self, password: str)
        Change password.

        Args: password (str) Real password string

    verify_password (self, password: str)
        Verify password.

        Args: password (str) Real password string

        Return: (bool) True if password matches else False

```

check_role (*self*, *role*: str)

Check if the user has a given role

static find_users ()

Get all the users

generate_token (*self*, *token_type*: str, *expiration*: int = 600)

Generate a token.

Args: token_type (str) Either TokenType.ACCESS_TOKEN, or TokenType.REFRESH_TOKEN expiration (int) Time to Live (TTL) in sec

Return: (str) Secured token

static verify_auth_token (*token*: str)

Verify token.

Args: token (str) Token

Return: (User) User object or None

class plynx.db.user.UserCollectionManager

User Manger

static find_user_by_name (*username*: str)

Find User.

Args: username (str) Username

Return: (User) User object or None

static find_user_by_email (*email*: str)

Find User.

Args: email (str) Email

Return: (User) User object or None

static get_users (*search*: str = "", *per_page*: int = 20, *offset*: int = 0)

Get a list of users

plynx.db.validation_error

Validation Error DB Object and utils

Module Contents

class plynx.db.validation_error.ValidationError

Bases: [plynx.db.db_object.DBObject](#)

Basic Validation Error class.

target :str

object_id :str

validation_code :str

children :List['ValidationError']

`plynx.db.worker_state`

Worker State DB Object and utils

Module Contents

class `plynx.db.worker_state.WorkerState`

Bases: `plynx.db.db_object.DBObject`

Worker statistics snapshot.

DB_COLLECTION

_id :ObjectId

worker_id :str =

host :str =

runs :List[Node]

kinds :List[str]

`plynx.db.worker_state.get_worker_states()` → List[WorkerState]

Get all of the workers with latest states

`plynx.demo`

Basic Operations for the demo.

Submodules

`plynx.demo.basic_functions`

PLynx Operations defined as python code. Using slightly more advanced functions than printing variables.

Module Contents

`plynx.demo.basic_functions.make_int(value)`

Pass Integer value as output.

`plynx.demo.basic_functions.make_enum(value)`

Pass Integer value as output.

`plynx.demo.basic_functions.example_func(x, y, coef)`

Math expression

`plynx.demo.basic_functions.sleep(x, timeout)`

Sleep for timeout sec and add 1.

`plynx.demo.basic_functions.error(x)`

Always raise exception

class `plynx.demo.basic_functions.Statefull`

Add 1 and keep the previous value

```
__call__(self, x)
```

```
plynx.demo.basic_functions.auto_run_disabled(x)
```

Auto run disabled for this node.

```
plynx.demo.basic_functions.GROUP
```

plynx.demo.hello_world

PLynx Operations defined as python code.

Module Contents

```
plynx.demo.hello_world.get_name(your_name)
```

Pass *your_name* parameter as output.

```
plynx.demo.hello_world.print_message(name)
```

Print greeting message.

```
plynx.demo.hello_world.GROUP
```

plynx.demo.types

PLynx Operations defined as python code. Produce basic types.

Module Contents

```
plynx.demo.types.print_any(value)
```

Format string to be html-friendly and print it.

```
plynx.demo.types.all_types()
```

Make basic values for each type.

```
plynx.demo.types.print_int(value)
```

Print input value.

```
plynx.demo.types.print_str(value)
```

Print input value.

```
plynx.demo.types.print_dict(value)
```

Print input value.

```
plynx.demo.types.print_float(value)
```

Print input value.

```
plynx.demo.types.print_bool(value)
```

Print input value.

```
plynx.demo.types.print_color(value)
```

Print input value.

```
plynx.demo.types.file_to_dict(value)
```

Transform file object to dict

```
plynx.demo.types.dict_to_file(value)
```

transform dict to file object

`plynx.demo.types.GROUP`

Package Contents

`plynx.demo.basic_group`

`plynx.demo.hello_group`

`plynx.demo.types_group`

`plynx.demo.COLLECTION`

`plynx.node`

PLynx API for generation user Nodes.

Submodules

`plynx.node.typing`

Standard PLYnx types

Module Contents

`plynx.node.typing.ANY = any`

`plynx.node.typing.STR_TO_CLASS :Dict[str, Union[Type, Callable]]`

`plynx.node.typing.CLASS_TO_STR :Dict[Union[Type, Callable], str]`

`plynx.node.typing.type_to_str (type_cls: Union[str, Type, Callable]) → str`
Standard type to PLYnx type

`plynx.node.utils`

General PLYnx utils for user-defined Operations.

Module Contents

`class plynx.node.utils.Group`

Collection of Operations

`title :str`

`items :list`

`to_dict (self)`

Dict representation

`plynx.node.utils.callable_to_function_location (callable_obj: Callable) → str`

Generate the location of the function

`plynx.node.utils.generate_key (node: Node) → str`
Generate hash of the node template.

`plynx.node.utils.func_or_group_to_dict (func_or_group: Union[Callable, Group])`
Recursive serializer

Package Contents

class `plynx.node.InputItem`
Input item abstraction

name :str

file_type :str

is_array :bool

min_count :int

to_dict (*self*)

Dict representation

class `plynx.node.OutputItem`
Output item abstraction

name :str

file_type :str

is_array :bool

min_count :int

to_dict (*self*)

Dict representation

class `plynx.node.ParamItem`
Parameter item abstraction

name :str

parameter_type :str

value :Any

widget :Optional[str]

to_dict (*self*)

Dict representation

class `plynx.node.PlynxParams`
Internal Plynx Node params

title :str

description :str

kind :str

node_type :str

auto_run_enabled :bool = True

inputs :List[InputItem]

params :List[ParamItem]

outputs :List[OutputItem]

`plynx.node.input` (*name=None, var_type=None, is_array=False, min_count=1*)
 Plynx Operation Input

`plynx.node.output` (*name=None, var_type=None, is_array=False, min_count=1*)
 Plynx Operation Output

`plynx.node.param` (*name=None, var_type=None, default=None, widget=""*)
 Plynx Operation Parameter

`plynx.node.operation` (*node_type=None, title=None, description="", kind=None, auto_run_enabled:*
bool = True)
 Plynx user-defined Operation

`plynx.node.parameter`

plynx.plugins

Subpackages

`plynx.plugins.executors`

Subpackages

`plynx.plugins.executors.python`

Submodules

`plynx.plugins.executors.python.dag`

An executor for the DAGs based on python backend.

Module Contents

`plynx.plugins.executors.python.dag.POOL_SIZE = 3`

`plynx.plugins.executors.python.dag.worker_main` (*job_run_queue:* *queue.Queue,*
job_complete_queue: queue.Queue)

Main threaded function that serves Operations.

class `plynx.plugins.executors.python.dag.DAGParallel` (*node: Node*)

Bases: `plynx.plugins.executors.dag.DAG`

Python DAG scheduler.

Args: `node_dict` (dict)

IS_GRAPH = True

GRAPH_ITERATION_SLEEP = 0

`pop_jobs` (*self*)

Get a set of nodes with satisfied dependencies

`_execute_node` (*self, node: Node*)

kill (*self*)

Force to kill the process.

The reason can be the fact it was working too long or parent executor canceled it.

finished (*self*)

Return True or False depending on the running status of the DAG.

class `plynx.plugins.executors.python.dag.DAG` (*node: Node*)

Bases: `plynx.plugins.executors.dag.DAG`

Base Executor class

IS_GRAPH :bool = True

kill (*self*)

Force to kill the process.

The reason can be the fact it was working too long or parent executor canceled it.

init_executor (*self*)

Initialize environment for the executor

clean_up_executor (*self*)

Clean up the environment created by executor

_apply_inputs (*self, node*)

run (*self, preview: bool = False*)

Main execution function.

class `plynx.plugins.executors.python.dag.ExecutorWithWebWorkerServer`

Bases: `plynx.plugins.executors.python.dag.DAG`

This executor is used for testing purposes only.

static request_task (*url, data*)

Send a request to the server

static fire_and_forget (*url, json*)

Fire and forget

launch (*self*)

Launch the executor

`plynx.plugins.executors.python.local`

Python Operation

Module Contents

`plynx.plugins.executors.python.local.DEFAULT_CMD = # Python Operation`

Explicit markup ends without a blank line; unexpected unindent.

The code of the operation have to be defined as a function ## A function must be named *operation*. # The arguments are named and must represent then inputs of the PLYnx Operation. # The function must return a dictionary that map to the outputs of PLYnx Operation

```
def operation(int_a, int_b):
```

Unexpected indentation.

```
    return {"sum": int_a + int_b}
```

```

plynx.plugins.executors.python.local.stateful_init_mutex
plynx.plugins.executors.python.local.stateful_class_registry
plynx.plugins.executors.python.local._resource_manager
plynx.plugins.executors.python.local.materialize_fn_or_cls (node: Node) → Callable
    Unpickle the function
plynx.plugins.executors.python.local.assign_outputs (node: Node, output_dict: Dict[str, Any])
    Apply output_dict to node's outputs.
class plynx.plugins.executors.python.local.redirect_to_plynx_logs (node: Node, std-
    out: str,
    stderr: str)
    Redirect stdout and stderr to standard PLYnx Outputs
    __enter__ (self)
    __exit__ (self, *args)
plynx.plugins.executors.python.local.prep_args (node: Node) → Dict[str, Any]
    Pythonize inputs and parameters
class plynx.plugins.executors.python.local.PythonNode
    Bases: plynx.plugins.executors.bases.PLYnxSyncExecutor
    Class is used as a placeholder for local python executor
    run (self, preview: bool = False)
    kill (self)
    classmethod get_default_node (cls, is_workflow: bool)
        Generate a new default Node for this executor

```

Submodules

plynx.plugins.executors.bases

Executors supporting PLYnx sync/async inference framework

Module Contents

```

plynx.plugins.executors.bases.run_cancellation_manager()
    Lazy RunCancellationManager object
class plynx.plugins.executors.bases.PLYnxAsyncExecutor
    Bases: plynx.base.executor.BaseExecutor
    Base Executor class that is using PLYnx Async Inference backend
    launch (self)
        Put the Node on the queue
    kill (self)

```

get_running_status (*self*)

Returns the status of the execution.

Async executions should sync with the remote and return the result immediately.

class `plynx.plugins.executors.bases.PLynxSyncExecutor`

Bases: `plynx.base.executor.BaseExecutor`

Base Executor class that is using PLYnx Sync Inference backend

launch (*self*)

Run the node now and return the status

class `plynx.plugins.executors.bases.PLynxAsyncExecutorWithDirectory` (*node*)

Bases: `plynx.plugins.executors.bases.PLynxAsyncExecutor`

Base Executor class that is using PLYnx Async Inference backend

init_executor (*self*)

Make tmp dir if it does not exist

clean_up_executor (*self*)

Remove tmp dir

plynx.plugins.executors.dag

A standard executor for DAGs.

Module Contents

`plynx.plugins.executors.dag._WAIT_STATUS_BEFORE_FAILED`

`plynx.plugins.executors.dag._ACTIVE_WAITING_TO_STOP`

`plynx.plugins.executors.dag.node_cache_manager()`

Lazy NodeCacheManager definition

class `plynx.plugins.executors.dag.DAG` (*node: Node*)

Bases: `plynx.plugins.executors.bases.PLynxAsyncExecutor`

Main graph scheduler.

Args: *node* (Node)

IS_GRAPH = True

GRAPH_ITERATION_SLEEP = 1

finished (*self*)

Return True or False depending on the running status of the DAG.

pop_jobs (*self*)

Get a set of nodes with satisfied dependencies

update_node (*self, node: Node*)

Update `node_running_status` and outputs if the state has changed.

_set_node_status (*self, node_id: ObjectId, node_running_status: str*)

static **_cacheable** (*node: Node*)

classmethod **get_default_node** (*cls, is_workflow: bool*)

```
_execute_node (self, node: Node)
run (self, preview: bool = False)
kill (self)
    Force to kill the process.

    The reason can be the fact it was working too long or parent executor canceled it.
validate (self, ignore_inputs: bool = True)
```

plynx.plugins.executors.local

Standard Executors that support running on local machine.

Module Contents

```
plynx.plugins.executors.local._resource_merger_func()
plynx.plugins.executors.local.prepare_parameters_for_python(parameters:
                                                            List[Parameter] →
                                                            Dict[str, Any])
```

Pythonize parameters

```
class plynx.plugins.executors.local._ResourceMerger(init_level_0: List[str],
                                                    init_level_1: List[str])
```

```
append (self, resource_dict: Dict[str, List[str]], resource_name: str, is_list: bool)
    Append values to the resource
```

```
get_dict (self)
    Return original dict.
```

Out: Dict

```
class plynx.plugins.executors.local.BaseBash(node: Optional[Node] = None)
    Bases: plynx.plugins.executors.bases.PLynxAsyncExecutorWithDirectory
    Base Executor that will use unix bash as a backend.
exec_script (self, script_location: str)
    Execute the script when inputs are initialized.
kill (self)
is_updated (self)
__getstate__ (self)
static _make_debug_text (text: str)
classmethod get_default_node (cls, is_workflow: bool)
_prepare_inputs (self, preview: bool = False)
_prepare_outputs (self, preview: bool = False)
_prepare_logs (self)
_get_script_fname (self, extension: str = '.sh')
```

```

    _prepare_parameters (self)
    _postprocess_outputs (self, outputs: Dict[str, str])
    _postprocess_logs (self)
    _extract_cmd_text (self)
    upload_logs (self, final: bool = False)
        Upload logs to the storage. When Final is False, only upload on update
    run (self, preview=False)

```

```

class plynx.plugins.executors.local.BashJinja2
    Bases: plynx.plugins.executors.local.BaseBash

    Local executor that uses jinja2 template to format a bash script.

    HELP_TEMPLATE = # Use templates: {}

    Explicit markup ends without a blank line; unexpected unindent.

    # For example {{{ ' {{{' }}} params['_timeout'] {{{ ' }}}' }}} or {{{ ' {{{' }}} inputs['abc'] {{{ ' }}}' }}}
    }}}

    run (self, preview: bool = False)

    classmethod get_default_node (cls, is_workflow: bool)

class plynx.plugins.executors.local.PythonNode (node: Optional[Node] = None)
    Bases: plynx.plugins.executors.local.BaseBash

    Local executor that uses python template to format a bash script.

    run (self, preview: bool = False)

    classmethod _get_arguments_string (cls, var_name: str, arguments: Dict[str, Any])

    static _pythonize (value: Any)

    classmethod get_default_node (cls, is_workflow: bool)

class plynx.plugins.executors.local.File
    Bases: plynx.plugins.executors.bases.PLynxAsyncExecutor

    Dummy executor that represents STATIC Operations.

    run (self, preview: bool = False)

    kill (self)

    classmethod get_default_node (cls, is_workflow: bool)

```

plynx.plugins.hubs

Submodules

plynx.plugins.hubs.collection

Plynx standard Hub based on the database of Operations

Module Contents

class `plynx.plugins.hubs.collection.CollectionHub` (*collection, operations*)

Bases: `plynx.base.hub.BaseHub`

Plynx standard Hub based on the database of Operations

search (*self, query: hub.Query*)

plynx.plugins.hubs.static_list

Plynx standard Hub based on the fixed list of Operations

Module Contents

`plynx.plugins.hubs.static_list.register_list_item` (*raw_item: Dict*) → Dict

Register a hub node (node or group) recursively in the memory.

`plynx.plugins.hubs.static_list._recursive_filter` (*search_parameters: Dict[str, str],
search_string: str, list_of_nodes:
List[Dict]*)

class `plynx.plugins.hubs.static_list.StaticListHub` (*list_nodes_path: str*)

Bases: `plynx.base.hub.BaseHub`

Plynx standard Hub based on the fixed list of Operations

search (*self, query: hub.Query*)

plynx.plugins.resources

Subpackages

plynx.plugins.resources.python

Submodules

plynx.plugins.resources.python.common

Commonly used Resource types and templates in python.

Module Contents

class `plynx.plugins.resources.python.common.Json`

Bases: `plynx.base.resource.BaseResource`

JSON file

DISPLAY_THUMBNAIL :bool = True

static preprocess_input (*value: Any*)

Resource_id to an object

```
static postprocess_output (value: Any)  
    Object to resource id  
  
classmethod preview (cls, preview_object: resource.PreviewObject)  
    Generate preview html body  
  
classmethod thumbnail (cls, output: Any)
```

Submodules

`plynx.plugins.resources.cloud_resources`

Resources that implement cloud abstractions

Module Contents

```
plynx.plugins.resources.cloud_resources.CLOUD_SERVICE_CONFIG  
  
class plynx.plugins.resources.cloud_resources.CloudStorage  
    Bases: plynx.base.resource.BaseResource  
    Storage Resource, i.e. S3 bucket  
  
    static prepare_input (filename: str, preview: bool = False)  
        Preprocess input  
  
    static prepare_output (filename: str, preview: bool = False)  
        Postprocess output  
  
    classmethod preview (cls, preview_object: resource.PreviewObject)  
        Preview resource
```

`plynx.plugins.resources.common`

Commonly used Resource types.

Module Contents

```
plynx.plugins.resources.common.WEB_CONFIG :WebConfig  
  
class plynx.plugins.resources.common.Raw  
    Bases: plynx.base.resource.BaseResource  
    Raw Resource that will be stored in jsonable format in the Node.  
  
    DISPLAY_RAW :bool = True  
  
class plynx.plugins.resources.common.RawInt  
    Bases: plynx.plugins.resources.common.Raw  
    Raw Resource that will store an integer in the Node.  
  
    static preprocess_input (value: Any)  
        Resource_id to an object
```

```

class plynx.plugins.resources.common.RawFloat
    Bases: plynx.plugins.resources.common.Raw

    Raw Resource that will store an integer in the Node.

    static preprocess_input (value: Any)
        Resource_id to an object

class plynx.plugins.resources.common.RawColor
    Bases: plynx.plugins.resources.common.Raw

    Raw Resource that will store an integer in the Node.

    static preprocess_input (value: Any)
        Resource_id to an object

class plynx.plugins.resources.common.File
    Bases: plynx.base.resource.BaseResource

    Raw Resource that will be stored in the file format in the Node.

class plynx.plugins.resources.common.PDF
    Bases: plynx.base.resource.BaseResource

    PDF file

    classmethod preview (cls, preview_object: resource.PreviewObject)
        Generate preview html body

class plynx.plugins.resources.common.Image
    Bases: plynx.base.resource.BaseResource

    Image file

    DISPLAY_THUMBNAIL :bool = True

    classmethod preview (cls, preview_object: resource.PreviewObject)
        Generate preview html body

    classmethod thumbnail (cls, output: Any)

class plynx.plugins.resources.common._BaseSeparated
    Bases: plynx.base.resource.BaseResource

    Base Separated file, i.e. csv, tsv

    SEPARATOR :Optional[str]

    _ROW_CLASSES :List[str] = ['even', 'odd']

    _NUM_ROW_CLASSES :int

    classmethod preview (cls, preview_object: resource.PreviewObject)
        Generate preview html body

class plynx.plugins.resources.common.CSV
    Bases: plynx.plugins.resources.common._BaseSeparated

    CSV file

    SEPARATOR :str = ,

class plynx.plugins.resources.common.TSV
    Bases: plynx.plugins.resources.common._BaseSeparated

    TSV file

```

```
SEPARATOR :str =

class plynx.plugins.resources.common.Json
    Bases: plynx.base.resource.BaseResource

    JSON file

    classmethod preview (cls, preview_object: resource.PreviewObject)
        Generate preview html body

class plynx.plugins.resources.common.Executable
    Bases: plynx.base.resource.BaseResource

    Executable file, i.e. bash or python

    static prepare_input (filename, preview: bool = False)
        Generate preview html body

class plynx.plugins.resources.common.Directory
    Bases: plynx.base.resource.BaseResource

    Directory file, i.e. zipfile

    static prepare_input (filename, preview: bool = False)
        Extract zip file

    static prepare_output (filename, preview: bool = False)
        Create output folder

    static postprocess_output (value: str)
        Compress folder to a zip file

    classmethod preview (cls, preview_object: resource.PreviewObject)
        Generate preview html body

plynx.plugins.resources.common.FILE_KIND = file
```

plynx.service

Submodules

plynx.service.cache

Main PLYnx cache service and utils

Module Contents

```
plynx.service.cache.OutputListTuple
plynx.service.cache.LIST_CACHE = list
plynx.service.cache.CLEAN_CACHE = clean
plynx.service.cache.MODES
plynx.service.cache.node_cache_manager :NodeCacheManager
plynx.service.cache.run_list_cache (start_datetime: Optional[datetime], end_datetime: date-
                                     time)
    Print all of the cache objects in a given time frame
```

`plynx.service.cache.run_clean_cache` (*start_datetime: Optional[datetime], end_datetime: datetime, yes: bool*)

Clean cache

`plynx.service.cache.run_cache` (*mode, start_datetime: str, end_datetime: str, yes: bool*)

Cache CLI entrypoint

`plynx.service.execute`

Main Plynx executor service and utils

Module Contents

`plynx.service.execute.run_execute` (*filename: str*)

Execute entrypoint. It materialize the Node based on file content and runs it.

`plynx.service.make_operations_meta`

Create metadata of operations

Module Contents

`plynx.service.make_operations_meta._enhance_list_item` (*raw_item: Dict*) → Dict

`plynx.service.make_operations_meta.run_make_operations_meta` (*collection_module, out*)

Make metadata

`plynx.service.users`

Main Plynx users service and utils

Module Contents

`plynx.service.users.LIST_USERS` = `list_users`

`plynx.service.users.CREATE_USER` = `create_user`

`plynx.service.users.ACTIVATE_USER` = `activate_user`

`plynx.service.users.DEACTIVATE_USER` = `deactivate_user`

`plynx.service.users.MODES`

`plynx.service.users.run_list_users` () → None

List all users

`plynx.service.users.run_create_user` (*email: Optional[str], username: Optional[str], password: Optional[str]*) → User

Create a user

`plynx.service.users.run_set_activation` (*username: Optional[str], value: bool*) → None

Set user active status

`plynx.service.users.run_users` (*mode: str, email: Optional[str] = None, username: Optional[str] = None, password: Optional[str] = ""*)

Users CLI entryptpoint

`plynx.service.worker`

Main Plynx worker service and utils

Module Contents

class `plynx.service.worker.Worker` (*worker_config: WorkerConfig, worker_id: Optional[str]*)

Worker main class.

On the high level Worker distributes Jobs over all available Workers and updates statuses of the Graphs in the database.

Worker performs several roles:

- Pull graphs in status READY from the database.
- Create Schedulers for each Graph.
- Populate the queue of the Jobs.
- Distribute Jobs accross Workers.
- Keep track of Job's statuses.
- Process CANCEL requests.
- Update Graph's statuses.
- Track worker status and last response.

SDB_STATUS_UPDATE_TIMEOUT :int = 1

WORKER_STATE_UPDATE_TIMEOUT :int = 1

serve_forever (*self*)

Run the worker.

execute_job (*self, executor: BaseExecutor*)

Run a single job in the executor

_run_db_status_update (*self*)

Syncing with the database.

_run_worker_state_update (*self*)

Syncing with the database.

stop (*self*)

Stop worker.

`plynx.service.worker.run_worker` (*worker_id: Optional[str] = None*)

Run worker daemon. It will run in the same thread.

`plynx.service.worker_server`

The serving logic of the worker

Module Contents

`plynx.service.worker_server.app`

`plynx.service.worker_server.logger`

class `plynx.service.worker_server.RunEnv`

Run environment or where the endpoint is running

HTTP = HTTP

PUBSUB = PUBSUB

`plynx.service.worker_server.execute_run()`

Execute a run with a given id

`plynx.service.worker_server.run_worker_server(verbose, endpoint_port: int)`

Run worker service

plynx.utils

Submodules

plynx.utils.common

Common utils

Module Contents

`plynx.utils.common.SearchParameter`

`plynx.utils.common.SEARCH_RGX`

`plynx.utils.common.TRUESES = ['true', 't']`

`plynx.utils.common.FALSESES = ['false', 'f']`

`plynx.utils.common.to_object_id(_id: Union[ObjectId, str, None]) → ObjectId`

Create ObjectId based on str, or return original value.

class `plynx.utils.common.JSONEncoder`

Bases: `json.JSONEncoder`

Handles some of the built in types

default (*self*, *o*)

`plynx.utils.common.zipdir(path: str, zf: zipfile.ZipFile)`

Walk in zip file

`plynx.utils.common.parse_search_string(search_string: str) → Tuple[Dict, str]`

Separate keywords fro msearch string

`plynx.utils.common.query_yes_no(question: str, default: str = 'yes') → bool`

Ask a yes/no question via input() and return their answer.

Args: question (str): String that is presented to the user. default (str): 'yes' or 'no' default value

The 'answer' return value is True for 'yes' or False for 'no'.

`plynx.utils.common.str_to_bool (val: Union[str, bool]) → bool`
Convert string value to boolean

`plynx.utils.common.update_dict_recursively (dest: Dict[Any, Any], donor: Dict[Any, Any]) → Dict[Any, Any]`
Update dictionary in place

`plynx.utils.config`

Global Plynx config

Module Contents

`plynx.utils.config.PLYNX_CONFIG_PATH :str`
`plynx.utils.config.DEFAULT_ICON :str = feathericons.x-square`
`plynx.utils.config.DEFAULT_COLOR :str = #ffffff`
`plynx.utils.config._CONFIG`
`plynx.utils.config.WorkerConfig`
`plynx.utils.config.MongoConfig`
`plynx.utils.config.StorageConfig`
`plynx.utils.config.AuthConfig`
`plynx.utils.config.WebConfig`
`plynx.utils.config.DemoConfig`
`plynx.utils.config.CloudServiceConfig`
`plynx.utils.config.ResourceConfig`
`plynx.utils.config.DummyOperationConfig`
`plynx.utils.config.OperationConfig`
`plynx.utils.config.HubConfig`
`plynx.utils.config.WorkflowConfig`
`plynx.utils.config.PluginsConfig`
`plynx.utils.config.IAMPoliciesConfig`
`plynx.utils.config.Config`
`plynx.utils.config._get_config() → Dict[str, Dict[str, Any]]`
Get global config
`plynx.utils.config.get_worker_config() → WorkerConfig`
Generate worker config
`plynx.utils.config.get_db_config() → MongoConfig`
Generate DB config
`plynx.utils.config.get_storage_config() → StorageConfig`
Generate Storage config

```

plynx.utils.config.get_auth_config() → AuthConfig
    Generate auth config

plynx.utils.config.get_web_config() → WebConfig
    Generate web config

plynx.utils.config.get_demo_config() → DemoConfig
    Generate web config

plynx.utils.config.get_cloud_service_config() → CloudServiceConfig
    Generate cloud config

plynx.utils.config.get_iam_policies_config() → IAMPoliciesConfig
    Generate IAM policies config

plynx.utils.config.get_plugins() → PluginsConfig
    Generate kind config

plynx.utils.config.get_config() → Config
    Generate full config

plynx.utils.config.set_parameter(levels: List[str], value: Any)
    Set global config parameter

    Args: levels (list): List of levels, i.e. ['mongodb', 'user'] value (value): Value of the parameter

plynx.utils.config._init_config()

```

plynx.utils.content

Create default empty Operations

Module Contents

```

plynx.utils.content.workflow_manager
plynx.utils.content.executor_manager
plynx.utils.content.create_template(user, kind, cmd, title, description, inputs=None, parameters=None, outputs=None)
plynx.utils.content.create_default_templates(user)

```

plynx.utils.db_connector

DB connector

Module Contents

```

plynx.utils.db_connector.PLYNX_DB = plynx
plynx.utils.db_connector._DB
plynx.utils.db_connector.init_indexes()
    Create DB indexes

plynx.utils.db_connector.get_db_connector()
    Create a connector lazily

```

```
plynx.utils.db_connector.check_connection()
```

Check DB connection

`plynx.utils.exceptions`

Standard Plynx Exceptions

Module Contents

exception `plynx.utils.exceptions.ExecutorNotFound`

Bases: `ImportError`

Executor not imported

exception `plynx.utils.exceptions.RegisterUserException` (*message: str, error_code: str*)

Bases: `Exception`

Failed to register the user

`plynx.utils.executor`

Utils to work with executors

Module Contents

`plynx.utils.executor.CONNECT_POST_TIMEOUT = 1.0`

`plynx.utils.executor.REQUESTS_TIMEOUT = 10`

`plynx.utils.executor.urljoin` (*base: str, postfix: str*) → `str`
Join urls in a reasonable way

`plynx.utils.executor.post_request` (*uri, data, num_retries=3*)
Make post request to the url

`plynx.utils.executor._update_node` (*node: plnx.db.node.Node*)
Update node in the database

`plynx.utils.executor.materialize_executor` (*node: Union[Dict[str, Any], plnx.db.node.Node]*) → `BaseExecutor`
Create a Node object from a dictionary

Parameters: `node`: dictionary representation of a Node or the node itself

Returns: `Executor`: Executor based on the kind of the Node and the config

class `plynx.utils.executor.TickThread` (*executor: BaseExecutor*)

This class is a Context Manager wrapper. It calls method `tick()` of the executor with a given interval

TICK_TIMEOUT : `int` = 1

`__enter__` (*self*)

Currently no meaning of returned class

`__exit__` (*self, type_cls, value, traceback_val*)

```
call_executor_tick (self)
    Run timed ticks
```

```
class plynx.utils.executor.DBJobExecutor (executor: BaseExecutor)
    Executes a single job in an executor and updates its status.
```

```
    run (self)
        Run the job in the executor
```

```
    kill (self)
        Kill the running job
```

plynx.utils.file_handler

Smart file handling

Module Contents

```
plynx.utils.file_handler._GLOBAL_STORAGE_CONFIG :Optional[StorageConfig]
```

```
plynx.utils.file_handler._get_global_storage_config() → StorageConfig
```

```
plynx.utils.file_handler.open (filename: str, mode: str = 'rt')
    Open file using internal configuration
```

```
plynx.utils.file_handler.get_file_stream (file_path: str, preview: bool = False,
                                          file_type=None) → BinaryIO
    Get file stream object (deprecated)
```

```
plynx.utils.file_handler.upload_file_stream (fp: BinaryIO, file_path: Optional[str] =
                                             None, seek: bool = True) → str
    Upload file stream to a given path (deprecated)
```

plynx.utils.hub_node_registry

Global Node Registry class

Module Contents

```
class plynx.utils.hub_node_registry.Registry
    The class keeps record of built-in Operations
```

```
    register_node (self, node: Node)
        Register Node globally
```

```
    find_nodes (self, function_locations: List[str])
        Find the Nodes
```

```
plynx.utils.hub_node_registry.registry
```

plynx.utils.logs

Logging utils

Module Contents

`plynx.utils.logs.set_logging_level` (*verbose: int, logger=None*)
Set logging level based on integer

`plynx.utils.node_utils`

This module contains utils related to `plynx.db.Node`, but not necessarily involved into DB structure

Module Contents

`plynx.utils.node_utils.node_collection_managers`

class `plynx.utils.node_utils._GraphVertex`
Used for internal purposes.

exception `plynx.utils.node_utils.GraphError`
Bases: `Exception`
Generic Graph topology exception

`plynx.utils.node_utils._generate_parameters_key` (*node: Node*) → str
Generate hash key based on parameters only.

Args: *node* (Node): Node object

Return: (str) Hash value

`plynx.utils.node_utils.node_inputs_and_params_are_identical` (*subnode: Node, other_subnode: Node*) → bool
Check if two nodes are identical in terms of inputs and parameters

`plynx.utils.node_utils.augment_node_with_cache` (*node: Node, other_node: Node*) → None
Augment the Node in templates with a Node in Run. Results will be stored in `_cached_node` fields of the subnodes and not applied directly.

`plynx.utils.node_utils.traverse_reversed` (*node: Node*)
Traverse the subnodes in a reversed from the topological order.

`plynx.utils.node_utils.traverse_in_order` (*node: Node*)
Traverse the subnodes in a topological order.

`plynx.utils.node_utils.arrange_auto_layout` (*node: Node, readonly: bool = False*)
Use heuristic to rearrange nodes.

`plynx.utils.node_utils.apply_cache` (*node: Node*)
Apply cache values to outputs and logs

`plynx.utils.node_utils.construct_new_run` (*node: Node, user_id*) → Tuple[Optional[Node], Node]
Create a new run based on a Node itself and the latest run as well.

`plynx.utils.node_utils.remove_auto_run_disabled` (*node: Node*)
Trim the subnodes the way that if there is no need to run a subnode and it is not auto runnable, ignore it.

`plynx.utils.node_utils.calc_status_to_node_ids` (*node: Optional[Node]*) → Dict[str, Set[ObjectId]]
Make a map `node_running_status` to list of ids.

`plynx.utils.node_utils.reset_nodes (node: Node)`

Reset statuses of the sub-nodes as well as logs and outputs

`plynx.utils.node_utils.traverse_left_join (node: Node, other_node: Node)`

Traverse two nodes in order and yield pairs of subnodes with the same `_id`.

`plynx.utils.plugin_manager`

Utils that materialize plugins

Module Contents

`plynx.utils.plugin_manager._isinstance_namedtuple (x: Any) → bool`

`plynx.utils.plugin_manager._as_dict (obj: Any)`

class `plynx.utils.plugin_manager._ResourceManager (plugins: PluginsConfig)`

class `plynx.utils.plugin_manager._ExecutorManager (plugins: PluginsConfig)`

class `plynx.utils.plugin_manager._OperationManager (plugins: PluginsConfig)`

class `plynx.utils.plugin_manager._HubManager (plugins: PluginsConfig)`

class `plynx.utils.plugin_manager._WorkflowManager (plugins: PluginsConfig)`

`plynx.utils.plugin_manager._plugins :PluginsConfig`

`plynx.utils.plugin_manager._RESOURCE_MANAGER`

`plynx.utils.plugin_manager._OPERATION_MANAGER`

`plynx.utils.plugin_manager._HUB_MANAGER`

`plynx.utils.plugin_manager._WORKFLOW_MANAGER`

`plynx.utils.plugin_manager._EXECUTOR_MANAGER`

`plynx.utils.plugin_manager._PLUGINS_DICT`

`plynx.utils.plugin_manager.get_resource_manager ()`

Generate Resource plugin structure

`plynx.utils.plugin_manager.get_operation_manager ()`

Generate Operation plugin structure

`plynx.utils.plugin_manager.get_hub_manager ()`

Generate Hub plugin structure

`plynx.utils.plugin_manager.get_workflow_manager ()`

Generate Workflow plugin structure

`plynx.utils.plugin_manager.get_executor_manager ()`

Generate Exectutor plugin structure

`plynx.utils.plugin_manager.get_plugins_dict ()`

Generate all of the plugins structure

`plynx.utils.thumbnails`

Thumbnail utils

Module Contents

`plynx.utils.thumbnails.get_thumbnail` (*output: plynx.db.node.Output*) → `Optional[str]`
Apply a single thumbnail

`plynx.utils.thumbnails.apply_thumbnails` (*node: plynx.db.node.Node*)
Fill thumbnail field of every subnode

`plynx.web`

Service endpoints.

Submodules

`plynx.web.common`

Common utils of the web service.

Module Contents

`plynx.web.common.app`

`plynx.web.common.logger`

`plynx.web.common.DEFAULT_EMAIL` =

`plynx.web.common.DEFAULT_USERNAME` = `default`

`plynx.web.common.DEFAULT_PASSWORD` =

`plynx.web.common._CONFIG`

`plynx.web.common.register_user` (*username: str, password: str, email: str, picture: str = "", is_oauth: bool = False, display_name: Optional[str] = None*)
→ `User`

Register a new user.

`plynx.web.common._init_default_user` ()

`plynx.web.common.verify_password` (*username_or_token: str, password: str*)
Verify password based on user

`plynx.web.common.authenticate` ()
Return 401 message

`plynx.web.common.requires_auth` (*f*)
Auth wrapper

`plynx.web.common.make_fail_response` (*message, **kwargs*)
Return basic fail response

`plynx.web.common.make_permission_denied` (*message: str = 'Permission denied'*)
Return permission error

`plynx.web.common.make_success_response` (*extra_response: Optional[Dict[str, Any]] = None*)
Return successful response

`plynx.web.common.handle_errors(f)`
Handle errors wrapper

`plynx.web.common.run_api(verbose)`
Run web service

`plynx.web.health`

Health check

Module Contents

`plynx.web.health.get_health_base()`
Health check

`plynx.web.health.get_health()`
Health check

`plynx.web.node`

All of the endpoints related to the Nodes or simial DB structures

Module Contents

`plynx.web.node.PAGINATION_QUERY_KEYS`

`plynx.web.node.node_collection_managers`

`plynx.web.node.run_cancellation_manager`

`plynx.web.node.operation_manager`

`plynx.web.node.hub_manager`

`plynx.web.node.workflow_manager`

`plynx.web.node.executor_manager`

`plynx.web.node.PLUGINS_DICT`

`plynx.web.node.post_search_nodes(collection: str)`
Create a search request in templates or runs

`plynx.web.node.get_nodes(collection: str, node_link: Optional[str] = None)`
Get the Node based on its ID or kind

`plynx.web.node.post_node(collection: str)`
Post a Node with an action

`plynx.web.resource`

All of the endpoints related to Resources

Module Contents

`plynx.web.resource.RESOURCE_TYPES`
`plynx.web.resource.get_resource(resource_id: str)`
Get the data of the resource
`plynx.web.resource.post_resource()`
Upload a new resource
`plynx.web.resource.upload_file()`
Upload file

`plynx.web.run`

All of the endpoints related to the the runs

Module Contents

`plynx.web.run.node_collection_manager`
`plynx.web.run.run_cancellation_manager`
`plynx.web.run.get_a_run()`
Find a certain run and return it
`plynx.web.run.pick_a_run()`
Find a single run and return it
`plynx.web.run.update_run()`
Update an entry in /runs collections
`plynx.web.run.get_run_cancelations()`
Ask the server if there is a cancelation

`plynx.web.state`

Endpoints responsible for the dashboard

Module Contents

`plynx.web.state.PLUGINS_DICT`
`plynx.web.state.worker_states()`
Get worker's states
`plynx.web.state.push_worker_state()`
Update the worker state

`plynx.web.user`

All of the endpoints related to Users

Module Contents

```
plynx.web.user.demo_user_manager
plynx.web.user.template_collection_manager
plynx.web.user.GOOGLE_CLIENT_ID
plynx.web.user.get_auth_token()
    Generate access and refresh tokens
plynx.web.user.get_user(username: str)
    Get user info by username
plynx.web.user.post_user()
    Update user info
plynx.web.user.post_register()
    Register a new user
plynx.web.user.post_register_with_oauth2()
    Register a new user
```

Package Contents

```
plynx.web.DEFAULT_EMAIL =
plynx.web.DEFAULT_PASSWORD =
plynx.web.DEFAULT_USERNAME = default
plynx.web.app
plynx.web.authenticate()
    Return 401 message
plynx.web.make_fail_response(message, **kwargs)
    Return basic fail response
plynx.web.requires_auth(f)
    Auth wrapper
plynx.web.run_api(verbose)
    Run web service
plynx.web.verify_password(username_or_token: str, password: str)
    Verify password based on user
```

1.11.1.2 Package Contents

```
plynx.__version__ = 1.11.1
```


p

- `plynx`, 37
- `plynx.base`, 37
- `plynx.base.executor`, 38
- `plynx.base.hub`, 39
- `plynx.base.resource`, 39
- `plynx.bin`, 40
- `plynx.bin.cli`, 40
- `plynx.constants`, 42
- `plynx.constants.collections`, 42
- `plynx.constants.node_enums`, 42
- `plynx.constants.parameter_types`, 44
- `plynx.constants.resource_enums`, 45
- `plynx.constants.users`, 45
- `plynx.constants.validation_enums`, 46
- `plynx.constants.web`, 47
- `plynx.db`, 51
- `plynx.db.db_object`, 51
- `plynx.db.demo_user_manager`, 52
- `plynx.db.node`, 52
- `plynx.db.node_cache`, 56
- `plynx.db.node_cache_manager`, 56
- `plynx.db.node_collection_manager`, 57
- `plynx.db.run_cancellation_manager`, 58
- `plynx.db.user`, 59
- `plynx.db.validation_error`, 60
- `plynx.db.worker_state`, 61
- `plynx.demo`, 61
- `plynx.demo.basic_functions`, 61
- `plynx.demo.hello_world`, 62
- `plynx.demo.types`, 62
- `plynx.node`, 63
- `plynx.node.typing`, 63
- `plynx.node.utils`, 63
- `plynx.plugins`, 65
- `plynx.plugins.executors`, 65
- `plynx.plugins.executors.bases`, 67
- `plynx.plugins.executors.dag`, 68
- `plynx.plugins.executors.local`, 69
- `plynx.plugins.executors.python`, 65
- `plynx.plugins.executors.python.dag`, 65
- `plynx.plugins.executors.python.local`, 66
- `plynx.plugins.hubs`, 70
- `plynx.plugins.hubs.collection`, 70
- `plynx.plugins.hubs.static_list`, 71
- `plynx.plugins.resources`, 71
- `plynx.plugins.resources.cloud_resources`, 72
- `plynx.plugins.resources.common`, 72
- `plynx.plugins.resources.python`, 71
- `plynx.plugins.resources.python.common`, 71
- `plynx.service`, 74
- `plynx.service.cache`, 74
- `plynx.service.execute`, 75
- `plynx.service.make_operations_meta`, 75
- `plynx.service.users`, 75
- `plynx.service.worker`, 76
- `plynx.service.worker_server`, 76
- `plynx.utils`, 77
- `plynx.utils.common`, 77
- `plynx.utils.config`, 78
- `plynx.utils.content`, 79
- `plynx.utils.db_connector`, 79
- `plynx.utils.exceptions`, 80
- `plynx.utils.executor`, 80
- `plynx.utils.file_handler`, 81
- `plynx.utils.hub_node_registry`, 81
- `plynx.utils.logs`, 81
- `plynx.utils.node_utils`, 82
- `plynx.utils.plugin_manager`, 83
- `plynx.utils.thumbnails`, 83
- `plynx.web`, 84
- `plynx.web.common`, 84
- `plynx.web.health`, 85
- `plynx.web.node`, 85
- `plynx.web.resource`, 85
- `plynx.web.run`, 86

`plynx.web.state`, [86](#)
`plynx.web.user`, [86](#)

Symbols

<code>_ACTIVE_WAITING_TO_STOP</code> (in module <code>pl-ynx.plugins.executors.dag</code>), 68	<code>_NON_CHANGEABLE_STATUSES</code> (<code>pl-ynx.constants.NodeRunningStatus</code> attribute), 48
<code>_AWAITING_STATUSES</code> (<code>pl-ynx.constants.NodeRunningStatus</code> attribute), 48	<code>_NON_CHANGEABLE_STATUSES</code> (<code>pl-ynx.constants.node_enums.NodeRunningStatus</code> attribute), 43
<code>_AWAITING_STATUSES</code> (<code>pl-ynx.constants.node_enums.NodeRunningStatus</code> attribute), 43	<code>_NUM_ROW_CLASSES</code> (<code>pl-ynx.plugins.resources.common._BaseSeparated</code> attribute), 73
<code>_BaseResource</code> (class in <code>plynx.db.node</code>), 53	<code>_OPERATION_MANAGER</code> (in module <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_BaseSeparated</code> (class in <code>pl-ynx.plugins.resources.common</code>), 73	<code>_OperationManager</code> (class in <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_CONFIG</code> (in module <code>plynx.utils.config</code>), 78	<code>_PLUGINS_DICT</code> (in module <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_CONFIG</code> (in module <code>plynx.web.common</code>), 84	<code>_PROPERTIES_TO_GET_FROM_SUBS</code> (in module <code>pl-ynx.db.node_collection_manager</code>), 57
<code>_DB</code> (in module <code>plynx.utils.db_connector</code>), 79	<code>_RESOURCE_MANAGER</code> (in module <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_DBObject</code> (class in <code>plynx.db.db_object</code>), 51	<code>_ROW_CLASSES</code> (<code>plynx.plugins.resources.common._BaseSeparated</code> attribute), 73
<code>_EXECUTOR_MANAGER</code> (in module <code>pl-ynx.utils.plugin_manager</code>), 83	<code>_ResourceManager</code> (class in <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_ExecutorManager</code> (class in <code>pl-ynx.utils.plugin_manager</code>), 83	<code>_ResourceMerger</code> (class in <code>pl-ynx.plugins.executors.local</code>), 69
<code>_FAILED_STATUSES</code> (<code>pl-ynx.constants.NodeRunningStatus</code> attribute), 48	<code>_SUCCEEDED_STATUSES</code> (<code>pl-ynx.constants.NodeRunningStatus</code> attribute), 48
<code>_FAILED_STATUSES</code> (<code>pl-ynx.constants.node_enums.NodeRunningStatus</code> attribute), 43	<code>_SUCCEEDED_STATUSES</code> (<code>pl-ynx.constants.node_enums.NodeRunningStatus</code> attribute), 43
<code>_FINISHED_STATUSES</code> (<code>pl-ynx.constants.NodeRunningStatus</code> attribute), 48	<code>_WAIT_STATUS_BEFORE_FAILED</code> (in module <code>pl-ynx.plugins.executors.dag</code>), 68
<code>_FINISHED_STATUSES</code> (<code>pl-ynx.constants.node_enums.NodeRunningStatus</code> attribute), 43	<code>_WORKFLOW_MANAGER</code> (in module <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_GLOBAL_STORAGE_CONFIG</code> (in module <code>pl-ynx.utils.file_handler</code>), 81	<code>_WorkflowManager</code> (class in <code>pl-ynx.utils.plugin_manager</code>), 83
<code>_GraphVertex</code> (class in <code>plynx.utils.node_utils</code>), 82	<code>__call__()</code> (<code>plynx.demo.basic_functions.Statefull</code> method), 61
<code>_HUB_MANAGER</code> (in module <code>pl-ynx.utils.plugin_manager</code>), 83	<code>__enter__()</code> (<code>plynx.plugins.executors.python.local.redirect_to_plynx_lo</code>
<code>_HubManager</code> (class in <code>plynx.utils.plugin_manager</code>), 83	

`method`), 67
`__enter__()` (`plynx.utils.executor.TickThread` `method`), 80
`__exit__()` (`plynx.plugins.executors.python.local.redirect_to_plynx_utils.file_handler`), 81
`method`), 67
`__exit__()` (`plynx.utils.executor.TickThread` `method`), 80
`__getstate__()` (`plynx.plugins.executors.local.BaseBash` `method`), 69
`__new__()` (`plynx.db.db_object.Meta` `class` `method`), 52
`__post_init__()` (`plynx.db.db_object.DBObject` `method`), 52
`__post_init__()` (`plynx.db.node.Input` `method`), 53
`__post_init__()` (`plynx.db.node.Parameter` `method`), 55
`__repr__()` (`plynx.db.db_object.DBObject` `method`), 52
`__str__()` (`plynx.db.db_object.DBObject` `method`), 52
`__version__` (*in* `module` `plynx`), 87
`_apply_inputs()` (`plynx.plugins.executors.python.dag.DAG` `method`), 66
`_as_dict()` (*in* `module` `plynx.utils.plugin_manager`), 83
`_cacheable()` (`plynx.plugins.executors.dag.DAG` `static` `method`), 68
`_cached_node` (`plynx.db.node.Node` `attribute`), 54
`_clone_update_in_place()` (*in* `module` `plynx.db.node`), 53
`_config` (*in* `module` `plynx.bin.cli`), 40
`_default_log()` (`plynx.db.node.Node` `static` `method`), 54
`_enhance_list_item()` (*in* `module` `plynx.service.make_operations_meta`), 75
`_execute_node()` (`plynx.plugins.executors.dag.DAG` `method`), 68
`_execute_node()` (`plynx.plugins.executors.python.dag.DAGParallel` `method`), 65
`_extract_cmd_text()` (`plynx.plugins.executors.local.BaseBash` `method`), 70
`_force_decode()` (*in* `module` `plynx.base.resource`), 40
`_generate_parameters_key()` (*in* `module` `plynx.utils.node_utils`), 82
`_get_arguments_string()` (`plynx.plugins.executors.local.PythonNode` `class` `method`), 70
`_get_config()` (*in* `module` `plynx.utils.config`), 78
`_get_custom_element()` (`plynx.db.node.Node` `method`), 54
`_get_default_by_type()` (*in* `module` `plynx.db.node`), 55
`_get_global_storage_config()` (*in* `module` `plynx.plugins.executors.local.BaseBash` `method`), 69
`_get_script_fname()` (`plynx.plugins.executors.local.BaseBash` `method`), 69
`_id` (`plynx.db.node.Node` `attribute`), 53
`_id` (`plynx.db.node_cache.NodeCache` `attribute`), 56
`_id` (`plynx.db.run_cancellation_manager.RunCancellation` `attribute`), 58
`_id` (`plynx.db.user.User` `attribute`), 59
`_id` (`plynx.db.worker_state.WorkerState` `attribute`), 61
`_id_generator()` (`plynx.db.demo_user_manager.DemoUserManager` `static` `method`), 52
`_init_config()` (*in* `module` `plynx.utils.config`), 79
`_init_default_user()` (*in* `module` `plynx.web.common`), 84
`_isinstance_namedtuple()` (*in* `module` `plynx.utils.plugin_manager`), 83
`_make_debug_text()` (`plynx.plugins.executors.local.BaseBash` `static` `method`), 69
`_make_query()` (`plynx.db.node_cache_manager.NodeCacheManager` `static` `method`), 57
`_plugins` (*in* `module` `plynx.utils.plugin_manager`), 83
`_postprocess_logs()` (`plynx.plugins.executors.local.BaseBash` `method`), 70
`_postprocess_outputs()` (`plynx.plugins.executors.local.BaseBash` `method`), 70
`_prepare_inputs()` (`plynx.plugins.executors.local.BaseBash` `method`), 69
`_prepare_logs()` (`plynx.plugins.executors.local.BaseBash` `method`), 69
`_prepare_outputs()` (`plynx.plugins.executors.local.BaseBash` `method`), 69
`_prepare_parameters()` (`plynx.plugins.executors.local.BaseBash` `method`), 69
`_pythonize()` (`plynx.plugins.executors.local.PythonNode` `static` `method`), 70
`_recursive_filter()` (*in* `module` `plynx.plugins.hubs.static_list`), 71
`_registry` (*in* `module` `plynx.db.db_object`), 51
`_resource_manager` (*in* `module` `plynx.plugins.executors.python.local`), 67
`_resource_merger_func()` (*in* `module` `plynx.plugins.executors.python.local`), 67

`ynx.plugins.executors.local`), 69
`_run_db_status_update()` (`pl-ynx.service.worker.Worker` method), 76
`_run_worker_state_update()` (`pl-ynx.service.worker.Worker` method), 76
`_set_node_status()` (`pl-ynx.plugins.executors.dag.DAG` method), 68
`_transplant_node()` (`pl-ynx.db.node_collection_manager.NodeCollectionManager` static method), 58
`_type` (`plynx.db.node.Node` attribute), 54
`_update_node()` (in module `plynx.utils.executor`), 80
`_update_node()` (`plynx.base.executor.BaseExecutor` method), 38
`_update_sub_nodes_fields()` (`pl-ynx.db.node_collection_manager.NodeCollectionManager` method), 58
`_value_is_valid()` (in module `plynx.db.node`), 55

A

`ACCESS_TOKEN` (`plynx.constants.TokenType` attribute), 50
`ACCESS_TOKEN` (`plynx.constants.users.TokenType` attribute), 46
`action` (`plynx.bin.cli.Arg` attribute), 40
`ACTIVATE_USER` (in module `plynx.service.users`), 75
`active` (`plynx.db.user.User` attribute), 59
`add_input_reference()` (`plynx.db.node.Input` method), 53
`all_types()` (in module `plynx.demo.types`), 62
`ANY` (in module `plynx.node.typing`), 63
`api()` (in module `plynx.bin.cli`), 41
`app` (in module `plynx.service.worker_server`), 77
`app` (in module `plynx.web`), 87
`app` (in module `plynx.web.common`), 84
`append()` (`plynx.plugins.executors.local._ResourceMerge` method), 69
`apply_cache()` (in module `plynx.utils.node_utils`), 82
`apply_properties()` (`plynx.db.node.Node` method), 54
`apply_thumbnails()` (in module `plynx.utils.thumbnails`), 84
`APPROVE` (`plynx.constants.node_enums.NodePostAction` attribute), 43
`APPROVE` (`plynx.constants.NodePostAction` attribute), 47
`Arg` (class in `plynx.bin.cli`), 40
`ARGS` (`plynx.bin.cli.CLIFactory` attribute), 41
`ARGS` (`plynx.bin.CLIFactory` attribute), 41
`arrange_auto_layout()` (in module `plynx.utils.node_utils`), 82
`assign_outputs()` (in module `plynx.plugins.executors.python.local`), 67
`augment_node_with_cache()` (in module `plynx.utils.node_utils`), 82
`AuthConfig` (in module `plynx.utils.config`), 78
`authenticate()` (in module `plynx.web`), 87
`authenticate()` (in module `plynx.web.common`), 84
`author` (`plynx.db.node.Node` attribute), 54
`auto_run` (`plynx.db.node.Node` attribute), 54
`auto_run_disabled()` (in module `plynx.demo.basic_functions`), 62
`auto_run_enabled` (`plynx.db.node.Node` attribute), 54
`auto_run_enabled` (`plynx.node.PlynxParams` attribute), 64
`auto_sync` (`plynx.db.node.Node` attribute), 54

B

`BaseBash` (class in `plynx.plugins.executors.local`), 69
`BaseExecutor` (class in `plynx.base.executor`), 38
`BaseHub` (class in `plynx.base.hub`), 39
`BaseResource` (class in `plynx.base.resource`), 40
`BashJinja2` (class in `plynx.plugins.executors.local`), 70
`basic_group` (in module `plynx.demo`), 63
`BLOCK` (`plynx.constants.validation_enums.ValidationTargetType` attribute), 46
`BLOCK` (`plynx.constants.ValidationTargetType` attribute), 50
`BOOL` (`plynx.constants.parameter_types.ParameterTypes` attribute), 44
`BOOL` (`plynx.constants.ParameterTypes` attribute), 49
`BUILT_IN_HUBS` (`plynx.constants.node_enums.NodeOrigin` attribute), 44
`BUILT_IN_HUBS` (`plynx.constants.NodeOrigin` attribute), 47

C

`cache()` (in module `plynx.bin.cli`), 41
`cache_url` (`plynx.db.node.Node` attribute), 54
`CachedNode` (class in `plynx.db.node`), 53
`calc_status_to_node_ids()` (in module `plynx.utils.node_utils`), 82
`call_executor_tick()` (`plynx.utils.executor.TickThread` method), 80
`callable_to_function_location()` (in module `plynx.node.utils`), 63
`CAN_CREATE_OPERATIONS` (`plynx.constants.IAMPolicies` attribute), 50
`CAN_CREATE_OPERATIONS` (`plynx.constants.users.IAMPolicies` attribute), 45
`CAN_CREATE_WORKFLOWS` (`plynx.constants.IAMPolicies` attribute), 50

CAN_CREATE_WORKFLOWS (pl-ynx.constants.users.IAMPolicies attribute), 45

CAN_MODIFY_OTHERS_WORKFLOWS (pl-ynx.constants.IAMPolicies attribute), 50

CAN_MODIFY_OTHERS_WORKFLOWS (pl-ynx.constants.users.IAMPolicies attribute), 45

CAN_RUN_WORKFLOWS (plynx.constants.IAMPolicies attribute), 50

CAN_RUN_WORKFLOWS (pl-ynx.constants.users.IAMPolicies attribute), 45

CAN_VIEW_OPERATIONS (pl-ynx.constants.IAMPolicies attribute), 50

CAN_VIEW_OPERATIONS (pl-ynx.constants.users.IAMPolicies attribute), 45

CAN_VIEW_OTHERS_OPERATIONS (pl-ynx.constants.IAMPolicies attribute), 50

CAN_VIEW_OTHERS_OPERATIONS (pl-ynx.constants.users.IAMPolicies attribute), 45

CAN_VIEW_OTHERS_WORKFLOWS (pl-ynx.constants.IAMPolicies attribute), 50

CAN_VIEW_OTHERS_WORKFLOWS (pl-ynx.constants.users.IAMPolicies attribute), 45

CAN_VIEW_WORKFLOWS (plynx.constants.IAMPolicies attribute), 50

CAN_VIEW_WORKFLOWS (pl-ynx.constants.users.IAMPolicies attribute), 45

CANCEL (plynx.constants.node_enums.NodePostAction attribute), 43

CANCEL (plynx.constants.NodePostAction attribute), 48

cancel_run() (plynx.db.run_cancellation_manager.RunCancellationManager static method), 58

CANCELED (plynx.constants.node_enums.NodeRunningStatus attribute), 42

CANCELED (plynx.constants.NodeRunningStatus attribute), 48

check_connection() (in module pl-ynx.utils.db_connector), 80

check_role() (plynx.db.user.User method), 59

children (plynx.db.validation_error.ValidationError attribute), 60

CLASS_TO_STR (in module plynx.node.typing), 63

ClassNotSavable, 51

CLEAN_CACHE (in module plynx.service.cache), 74

clean_up() (plynx.db.node_cache_manager.NodeCacheManager static method), 57

clean_up_executor() (pl-ynx.base.executor.BaseExecutor method), 39

clean_up_executor() (pl-ynx.plugins.executors.bases.PLynxAsyncExecutorWithDirectory method), 68

clean_up_executor() (pl-ynx.plugins.executors.python.dag.DAG method), 66

CLIFactory (class in plynx.bin), 41

CLIFactory (class in plynx.bin.cli), 41

CLONE (plynx.constants.node_enums.NodePostAction attribute), 43

CLONE (plynx.constants.NodePostAction attribute), 47

clone() (plynx.db.node.Node method), 54

CLOUD_INPUT (plynx.constants.NodeResources attribute), 49

CLOUD_INPUT (plynx.constants.resource_enums.NodeResources attribute), 45

CLOUD_OUTPUT (plynx.constants.NodeResources attribute), 49

CLOUD_OUTPUT (plynx.constants.resource_enums.NodeResources attribute), 45

CLOUD_SERVICE_CONFIG (in module pl-ynx.plugins.resources.cloud_resources), 72

CloudServiceConfig (in module plynx.utils.config), 78

CloudStorage (class in pl-ynx.plugins.resources.cloud_resources), 72

CODE (plynx.constants.parameter_types.ParameterTypes attribute), 44

CODE (plynx.constants.ParameterTypes attribute), 49

code_function_location (plynx.db.node.Node attribute), 54

code_hash (plynx.db.node.Node attribute), 54

COLLECTION (in module plynx.demo), 63

CollectionManager (class in pl-ynx.plugins.hubs.collection), 71

collections (class in plynx.constants), 47

Collections (class in plynx.constants.collections), 42

COLOR (plynx.constants.parameter_types.ParameterTypes attribute), 44

COLOR (plynx.constants.ParameterTypes attribute), 49

Config (in module plynx.utils.config), 78

CONNECT_POST_TIMEOUT (in module pl-ynx.utils.executor), 80

construct_new_run() (in module pl-ynx.utils.node_utils), 82

copy() (plynx.db.db_object._DBObject method), 51

create_default_templates() (in module pl-ynx.utils.content), 79

create_demo_templates() (pl-ynx.db.demo_user_manager.DemoUserManager static method), 52

[create_demo_user\(\)](#) (*pl-ynx.db.demo_user_manager.DemoUserManager static method*), 52
[CREATE_RUN](#) (*pl-ynx.constants.node_enums.NodePostAction attribute*), 43
[CREATE_RUN](#) (*pl-ynx.constants.NodePostAction attribute*), 47
[CREATE_RUN_FROM_SCRATCH](#) (*pl-ynx.constants.node_enums.NodePostAction attribute*), 43
[CREATE_RUN_FROM_SCRATCH](#) (*pl-ynx.constants.NodePostAction attribute*), 47
[create_template\(\)](#) (*in module plynx.utils.content*), 79
[CREATE_USER](#) (*in module plynx.service.users*), 75
[CREATED](#) (*pl-ynx.constants.node_enums.NodeRunningStatus attribute*), 42
[CREATED](#) (*pl-ynx.constants.node_enums.NodeStatus attribute*), 43
[CREATED](#) (*pl-ynx.constants.NodeRunningStatus attribute*), 48
[CREATED](#) (*pl-ynx.constants.NodeStatus attribute*), 49
[CSV](#) (*class in plynx.plugins.resources.common*), 73

D

[DAG](#) (*class in plynx.plugins.executors.dag*), 68
[DAG](#) (*class in plynx.plugins.executors.python.dag*), 66
[DAGParallel](#) (*class in pl-ynx.plugins.executors.python.dag*), 65
[DB](#) (*pl-ynx.constants.node_enums.NodeOrigin attribute*), 44
[DB](#) (*pl-ynx.constants.NodeOrigin attribute*), 47
[DB_COLLECTION](#) (*pl-ynx.db.db_object.DBObject attribute*), 51
[DB_COLLECTION](#) (*pl-ynx.db.node.Node attribute*), 53
[DB_COLLECTION](#) (*pl-ynx.db.node_cache.NodeCache attribute*), 56
[DB_COLLECTION](#) (*pl-ynx.db.run_cancellation_manager.RunCancellation attribute*), 58
[DB_COLLECTION](#) (*pl-ynx.db.user.User attribute*), 59
[DB_COLLECTION](#) (*pl-ynx.db.worker_state.WorkerState attribute*), 61
[DBJobExecutor](#) (*class in plynx.utils.executor*), 81
[DBObject](#) (*class in plynx.db.db_object*), 52
[DBObjectNotFound](#), 51
[DBObjectType](#) (*in module plynx.db.db_object*), 51
[DEACTIVATE_USER](#) (*in module plynx.service.users*), 75
[default](#) (*pl-ynx.bin.cli.Arg attribute*), 40
[default\(\)](#) (*pl-ynx.utils.common.JSONEncoder method*), 77
[DEFAULT_CMD](#) (*in module pl-ynx.plugins.executors.python.local*), 66
[DEFAULT_COLOR](#) (*in module plynx.utils.config*), 78
[DEFAULT_EMAIL](#) (*in module plynx.web*), 87
[DEFAULT_EMAIL](#) (*in module plynx.web.common*), 84
[DEFAULT_ICON](#) (*in module plynx.utils.config*), 78
[DEFAULT_PASSWORD](#) (*in module plynx.web*), 87
[DEFAULT_PASSWORD](#) (*in module plynx.web.common*), 84
[DEFAULT_POLICIES](#) (*in module plynx.db.user*), 59
[DEFAULT_USERNAME](#) (*in module plynx.web*), 87
[DEFAULT_USERNAME](#) (*in module plynx.web.common*), 84
[demo_config](#) (*in module plynx.db.node_cache*), 56
[demo_config](#) (*pl-ynx.db.demo_user_manager.DemoUserManager attribute*), 52
[demo_user_manager](#) (*in module plynx.web.user*), 87
[DemoConfig](#) (*in module plynx.utils.config*), 78
[DemoUserManager](#) (*class in pl-ynx.db.demo_user_manager*), 52
[DEPRECATE](#) (*pl-ynx.constants.node_enums.NodePostAction attribute*), 43
[DEPRECATE](#) (*pl-ynx.constants.NodePostAction attribute*), 47
[DEPRECATED](#) (*pl-ynx.constants.node_enums.NodeStatus attribute*), 43
[DEPRECATED](#) (*pl-ynx.constants.NodeStatus attribute*), 49
[DEPRECATED_NODE](#) (*pl-ynx.constants.validation_enums.ValidationCode attribute*), 46
[DEPRECATED_NODE](#) (*pl-ynx.constants.ValidationCode attribute*), 50
[description](#) (*pl-ynx.db.node.Node attribute*), 54
[description](#) (*pl-ynx.node.PlynxParams attribute*), 64
[dict_to_file\(\)](#) (*in module plynx.demo.types*), 62
[Directory](#) (*class in plynx.plugins.resources.common*), 74
[display_name](#) (*pl-ynx.db.user.UserSettings attribute*), 59
[DISPLAY_RAW](#) (*pl-ynx.base.resource.BaseResource attribute*), 40
[DISPLAY_RAW](#) (*pl-ynx.plugins.resources.common.Raw attribute*), 72
[DISPLAY_THUMBNAIL](#) (*pl-ynx.base.resource.BaseResource attribute*), 40
[DISPLAY_THUMBNAIL](#) (*pl-ynx.plugins.resources.common.Image attribute*), 73
[DISPLAY_THUMBNAIL](#) (*pl-ynx.plugins.resources.python.common.Json attribute*), 71
[Dummy](#) (*class in plynx.base.executor*), 39
[DummyOperationConfig](#) (*in module pl-*

`ynx.utils.config`), 78

E

`email` (`plynx.db.user.User` attribute), 59

`EMAIL_ALREADY_EXISTS` (`plynx.constants.RegisterUserExceptionCode` attribute), 50

`EMAIL_ALREADY_EXISTS` (`plynx.constants.users.RegisterUserExceptionCode` attribute), 46

`EMPTY_GRAPH` (`plynx.constants.validation_enums.ValidationCode` attribute), 46

`EMPTY_GRAPH` (`plynx.constants.ValidationCode` attribute), 50

`EMPTY_PASSWORD` (`plynx.constants.RegisterUserExceptionCode` attribute), 50

`EMPTY_PASSWORD` (`plynx.constants.users.RegisterUserExceptionCode` attribute), 46

`EMPTY_USERNAME` (`plynx.constants.RegisterUserExceptionCode` attribute), 50

`EMPTY_USERNAME` (`plynx.constants.users.RegisterUserExceptionCode` attribute), 46

`ENUM` (`plynx.constants.parameter_types.ParameterTypes` attribute), 44

`ENUM` (`plynx.constants.ParameterTypes` attribute), 49

`error()` (in module `plynx.demo.basic_functions`), 61

`example_func()` (in module `plynx.demo.basic_functions`), 61

`exec_script()` (`plynx.plugins.executors.local.BaseBash` method), 69

`Executable` (class in `plynx.plugins.resources.common`), 74

`execute()` (in module `plynx.bin.cli`), 41

`execute_job()` (`plynx.service.worker.Worker` method), 76

`execute_run()` (in module `plynx.service.worker_server`), 77

`executor_manager` (in module `plynx.utils.content`), 79

`executor_manager` (in module `plynx.web.node`), 85

`ExecutorNotFound`, 80

`ExecutorWithWebWorkerServer` (class in `plynx.plugins.executors.python.dag`), 66

F

`FAILED` (`plynx.constants.node_enums.NodePostStatus` attribute), 43

`FAILED` (`plynx.constants.node_enums.NodeRunningStatus` attribute), 42

`FAILED` (`plynx.constants.NodePostStatus` attribute), 48

`FAILED` (`plynx.constants.NodeRunningStatus` attribute), 48

`FAILED` (`plynx.constants.ResponseStatus` attribute), 51

`FAILED` (`plynx.constants.web.ResponseStatus` attribute), 47

`FAILED_WAITING` (`plynx.constants.node_enums.NodeRunningStatus` attribute), 42

`FAILED_WAITING` (`plynx.constants.NodeRunningStatus` attribute), 48

`FALSES` (in module `plynx.utils.common`), 77

`File` (class in `plynx.plugins.executors.local`), 70

`File` (class in `plynx.plugins.resources.common`), 73

`FILE_KIND` (in module `plynx.plugins.resources.common`), 74

`file_to_dict()` (in module `plynx.demo.types`), 62

`file_type` (`plynx.db.node._BaseResource` attribute), 53

`file_type` (`plynx.node.InputItem` attribute), 64

`file_type` (`plynx.node.OutputItem` attribute), 64

`find_nodes()` (`plynx.utils.hub_node_registry.Registry` method), 81

`find_user_by_email()` (`plynx.db.user.UserCollectionManager` static method), 60

`find_user_by_name()` (`plynx.db.user.UserCollectionManager` static method), 60

`find_users()` (`plynx.db.user.User` static method), 60

`finished()` (`plynx.plugins.executors.dag.DAG` method), 68

`finished()` (`plynx.plugins.executors.python.dag.DAGParallel` method), 66

`fire_and_forget()` (`plynx.plugins.executors.python.dag.ExecutorWithWebWorkerServer` static method), 66

`flags` (`plynx.bin.cli.Arg` attribute), 40

`FLOAT` (`plynx.constants.parameter_types.ParameterTypes` attribute), 44

`FLOAT` (`plynx.constants.ParameterTypes` attribute), 49

`from_dict()` (`plynx.db.db_object._DBObject` class method), 52

`func_or_group_to_dict()` (in module `plynx.node.utils`), 64

G

`GENERATE_CODE` (`plynx.constants.node_enums.NodePostAction` attribute), 43

`GENERATE_CODE` (`plynx.constants.NodePostAction` attribute), 48

`generate_key()` (in module `plynx.node.utils`), 63

`generate_key()` (*plynx.db.node_cache.NodeCache static method*), 56
`generate_token()` (*plynx.db.user.User method*), 60
`get()` (*plynx.db.node_cache_manager.NodeCacheManager static method*), 57
`get_a_run()` (*in module plynx.web.run*), 86
`get_auth_config()` (*in module plynx.utils.config*), 78
`get_auth_token()` (*in module plynx.web.user*), 87
`get_class()` (*in module plynx.db.db_object*), 51
`get_cloud_service_config()` (*in module plynx.utils.config*), 79
`get_config()` (*in module plynx.utils.config*), 79
`get_db_config()` (*in module plynx.utils.config*), 78
`get_db_connector()` (*in module plynx.utils.db_connector*), 79
`get_db_node()` (*plynx.db.node_collection_manager.NodeCollectionManager method*), 58
`get_db_object()` (*plynx.db.node_collection_manager.NodeCollectionManager method*), 58
`get_db_objects()` (*plynx.db.node_collection_manager.NodeCollectionManager method*), 57
`get_db_objects_by_ids()` (*plynx.db.node_collection_manager.NodeCollectionManager method*), 57
`get_default_node()` (*plynx.base.executor.BaseExecutor class method*), 38
`get_default_node()` (*plynx.base.executor.Dummy class method*), 39
`get_default_node()` (*plynx.plugins.executors.dag.DAG class method*), 68
`get_default_node()` (*plynx.plugins.executors.local.BaseBash class method*), 69
`get_default_node()` (*plynx.plugins.executors.local.BashJinja2 class method*), 70
`get_default_node()` (*plynx.plugins.executors.local.File class method*), 70
`get_default_node()` (*plynx.plugins.executors.local.PythonNode class method*), 70
`get_default_node()` (*plynx.plugins.executors.python.local.PythonNode class method*), 67
`get_demo_config()` (*in module plynx.utils.config*), 79
`get_dict()` (*plynx.plugins.executors.local.ResourceMerge method*), 69
`get_executor_manager()` (*in module plynx.utils.plugin_manager*), 83
`get_file_stream()` (*in module plynx.utils.file_handler*), 81
`get_health()` (*in module plynx.web.health*), 85
`get_health_base()` (*in module plynx.web.health*), 85
`get_hub_manager()` (*in module plynx.utils.plugin_manager*), 83
`get_iam_policies_config()` (*in module plynx.utils.config*), 79
`get_input_by_name()` (*plynx.db.node.Node method*), 54
`get_list()` (*plynx.db.node_cache_manager.NodeCacheManager static method*), 57
`get_log_by_name()` (*plynx.db.node.Node method*), 62
`get_name()` (*in module plynx.demo.hello_world*), 62
`get_nodes()` (*in module plynx.web.node*), 85
`operation_manager()` (*in module plynx.utils.plugin_manager*), 83
`get_output_by_name()` (*plynx.db.node.Node method*), 55
`get_parameter_by_name()` (*plynx.db.node.Node method*), 55
`get_parameter_by_name_safe()` (*plynx.db.node.Node method*), 55
`get_parser()` (*in module plynx.bin.cli*), 41
`get_parser()` (*plynx.bin.cli.CLIFactory class method*), 41
`get_parser()` (*plynx.bin.CLIFactory class method*), 41
`get_plugins()` (*in module plynx.utils.config*), 79
`get_plugins_dict()` (*in module plynx.utils.plugin_manager*), 83
`get_resource()` (*in module plynx.web.resource*), 86
`get_resource_manager()` (*in module plynx.utils.plugin_manager*), 83
`get_run_cancelations()` (*in module plynx.web.run*), 86
`get_run_cancellations()` (*plynx.db.run_cancellation_manager.RunCancellationManager static method*), 59
`get_running_status()` (*plynx.base.executor.BaseExecutor method*), 38
`get_running_status()` (*plynx.plugins.executors.bases.PLynxAsyncExecutor method*), 67
`get_storage_config()` (*in module plynx.utils.config*), 78
`get_sub_nodes()` (*plynx.db.node.Node method*), 55
`get_thumbnail()` (*in module pl-*

- `ynx.utils.thumbnails`), 84
 - `get_user()` (in module `plynx.web.user`), 87
 - `get_users()` (`plynx.db.user.UserCollectionManager` static method), 60
 - `get_web_config()` (in module `plynx.utils.config`), 79
 - `get_worker_config()` (in module `plynx.utils.config`), 78
 - `get_worker_states()` (in module `plynx.db.worker_state`), 61
 - `get_workflow_manager()` (in module `plynx.utils.plugin_manager`), 83
 - `GOOGLE_CLIENT_ID` (in module `plynx.web.user`), 87
 - `GRAPH` (`plynx.constants.validation_enums.ValidationTargetType` attribute), 46
 - `GRAPH` (`plynx.constants.ValidationTargetType` attribute), 51
 - `GRAPH_ITERATION_SLEEP` (`plynx.plugins.executors.dag.DAG` attribute), 68
 - `GRAPH_ITERATION_SLEEP` (`plynx.plugins.executors.python.dag.DAGParallel` attribute), 65
 - `GraphError`, 82
 - `Group` (class in `plynx.node.utils`), 63
 - `GROUP` (in module `plynx.demo.basic_functions`), 62
 - `GROUP` (in module `plynx.demo.hello_world`), 62
 - `GROUP` (in module `plynx.demo.types`), 62
- ## H
- `handle_errors()` (in module `plynx.web.common`), 84
 - `hash_password()` (`plynx.db.user.User` method), 59
 - `hello_group` (in module `plynx.demo`), 63
 - `help` (`plynx.bin.cli.Arg` attribute), 40
 - `HELP_TEMPLATE` (`plynx.plugins.executors.local.BashJinja2` attribute), 70
 - `host` (`plynx.db.worker_state.WorkerState` attribute), 61
 - `HTTP` (`plynx.service.worker_server.RunEnv` attribute), 77
 - `hub_manager` (in module `plynx.web.node`), 85
 - `HUB_NODE_REGISTRY` (`plynx.constants.Collections` attribute), 47
 - `HUB_NODE_REGISTRY` (`plynx.constants.collections.Collections` attribute), 42
 - `HubConfig` (in module `plynx.utils.config`), 78
 - `HubSearchParams` (class in `plynx.constants`), 49
 - `HubSearchParams` (class in `plynx.constants.resource_enums`), 45
- ## I
- `IAMPolicies` (class in `plynx.constants`), 50
 - `IAMPolicies` (class in `plynx.constants.users`), 45
 - `IAMPoliciesConfig` (in module `plynx.utils.config`), 78
 - `IGNORED_CACHE_PARAMETERS` (in module `plynx.constants`), 47
 - `IGNORED_CACHE_PARAMETERS` (in module `plynx.constants.node_enums`), 44
 - `Image` (class in `plynx.plugins.resources.common`), 73
 - `IN_DEPENDENTS` (`plynx.constants.validation_enums.ValidationCode` attribute), 46
 - `IN_DEPENDENTS` (`plynx.constants.ValidationCode` attribute), 50
 - `IN_QUEUE` (`plynx.constants.node_enums.NodeRunningStatus` attribute), 42
 - `IN_QUEUE` (`plynx.constants.NodeRunningStatus` attribute), 48
 - `index` (`plynx.db.node.ParameterEnum` attribute), 55
 - `init_executor()` (`plynx.base.executor.BaseExecutor` method), 38
 - `init_executor()` (`plynx.plugins.executors.bases.PLynxAsyncExecutorWithDirectory` method), 68
 - `init_executor()` (`plynx.plugins.executors.python.dag.DAG` method), 66
 - `init_indexes()` (in module `plynx.utils.db_connector`), 79
 - `Input` (class in `plynx.db.node`), 53
 - `INPUT` (`plynx.constants.node_enums.SpecialNodeId` attribute), 44
 - `INPUT` (`plynx.constants.NodeResources` attribute), 49
 - `INPUT` (`plynx.constants.resource_enums.NodeResources` attribute), 45
 - `INPUT` (`plynx.constants.SpecialNodeId` attribute), 49
 - `INPUT` (`plynx.constants.validation_enums.ValidationTargetType` attribute), 46
 - `INPUT` (`plynx.constants.ValidationTargetType` attribute), 51
 - `input()` (in module `plynx.node`), 65
 - `INPUT_FILE_TYPE` (`plynx.constants.HubSearchParams` attribute), 49
 - `INPUT_FILE_TYPE` (`plynx.constants.resource_enums.HubSearchParams` attribute), 45
 - `input_references` (`plynx.db.node.Input` attribute), 53
 - `InputItem` (class in `plynx.node`), 64
 - `InputReference` (class in `plynx.db.node`), 53
 - `inputs` (`plynx.db.node.Node` attribute), 54
 - `inputs` (`plynx.node.PlynxParams` attribute), 64
 - `instantiate()` (`plynx.db.node_cache.NodeCache` static method), 56

- INT (*plynx.constants.parameter_types.ParameterTypes attribute*), 44
- INT (*plynx.constants.ParameterTypes attribute*), 49
- INVALID_EMAIL (*plynx.constants.RegisterUserExceptionCode attribute*), 50
- INVALID_EMAIL (*plynx.constants.users.RegisterUserExceptionCode attribute*), 46
- INVALID_LENGTH_OF_USERNAME (*plynx.constants.RegisterUserExceptionCode attribute*), 50
- INVALID_LENGTH_OF_USERNAME (*plynx.constants.users.RegisterUserExceptionCode attribute*), 46
- INVALID_VALUE (*plynx.constants.validation_enums.ValidationCode attribute*), 46
- INVALID_VALUE (*plynx.constants.ValidationCode attribute*), 50
- IS_ADMIN (*plynx.constants.IAMPolicies attribute*), 50
- IS_ADMIN (*plynx.constants.users.IAMPolicies attribute*), 45
- is_array (*plynx.db.node._BaseResource attribute*), 53
- is_array (*plynx.node.InputItem attribute*), 64
- is_array (*plynx.node.OutputItem attribute*), 64
- is_failed() (*plynx.constants.node_enums.NodeRunningStatus static method*), 43
- is_failed() (*plynx.constants.NodeRunningStatus static method*), 48
- is_finished() (*plynx.constants.node_enums.NodeRunningStatus static method*), 43
- is_finished() (*plynx.constants.NodeRunningStatus static method*), 48
- IS_GRAPH (*plynx.base.executor.BaseExecutor attribute*), 38
- IS_GRAPH (*plynx.plugins.executors.dag.DAG attribute*), 68
- IS_GRAPH (*plynx.plugins.executors.python.dag.DAG attribute*), 66
- IS_GRAPH (*plynx.plugins.executors.python.dag.DAGParallel attribute*), 65
- is_non_changeable() (*plynx.constants.node_enums.NodeRunningStatus static method*), 43
- is_non_changeable() (*plynx.constants.NodeRunningStatus static method*), 48
- is_succeeded() (*plynx.constants.node_enums.NodeRunningStatus static method*), 43
- is_succeeded() (*plynx.constants.NodeRunningStatus static method*), 49
- method), 48
- is_updated() (*plynx.base.executor.BaseExecutor method*), 38
- is_updated() (*plynx.plugins.executors.local.BaseBash method*), 69
- items (*plynx.node.utils.Group attribute*), 63
- ## J
- Json (*class in plynx.plugins.resources.common*), 74
- Json (*class in plynx.plugins.resources.python.common*), 71
- JSONEncoder (*class in plynx.utils.common*), 77
- JWT_ENCODE_ALGORITHM (*in module plynx.db.user*), 59
- ## K
- key (*plynx.db.node_cache.NodeCache attribute*), 56
- kill() (*plynx.base.executor.BaseExecutor method*), 38
- kill() (*plynx.base.executor.Dummy method*), 39
- kill() (*plynx.plugins.executors.bases.PLynxAsyncExecutor method*), 67
- kill() (*plynx.plugins.executors.dag.DAG method*), 69
- kill() (*plynx.plugins.executors.local.BaseBash method*), 69
- kill() (*plynx.plugins.executors.local.File method*), 70
- kill() (*plynx.plugins.executors.python.dag.DAG method*), 66
- kill() (*plynx.plugins.executors.python.dag.DAGParallel method*), 65
- kill() (*plynx.plugins.executors.python.local.PythonNode method*), 67
- kill() (*plynx.utils.executor.DBJobExecutor method*), 81
- kind (*plynx.db.node.Node attribute*), 54
- kind (*plynx.node.PlynxParams attribute*), 64
- kinds (*plynx.db.worker_state.WorkerState attribute*), 61
- ## L
- latest_run_id (*plynx.db.node.Node attribute*), 54
- launch() (*plynx.base.executor.BaseExecutor method*), 38
- launch() (*plynx.plugins.executors.bases.PLynxAsyncExecutor method*), 67
- launch() (*plynx.plugins.executors.bases.PLynxSyncExecutor method*), 68
- launch() (*plynx.plugins.executors.python.dag.ExecutorWithWebWorkerS method*), 66
- levels (*plynx.bin.cli.Arg attribute*), 40
- LIST_CACHE (*in module plynx.service.cache*), 74
- LIST_INT (*plynx.constants.parameter_types.ParameterTypes attribute*), 44
- LIST_INT (*plynx.constants.ParameterTypes attribute*), 49

LIST_NODE (*plynx.constants.parameter_types.ParameterTypes* attribute), 44

LIST_NODE (*plynx.constants.ParameterTypes* attribute), 49

LIST_STR (*plynx.constants.parameter_types.ParameterTypes* attribute), 44

LIST_STR (*plynx.constants.ParameterTypes* attribute), 49

LIST_USERS (in module *plynx.service.users*), 75

load() (*plynx.db.db_object.DBObject* class method), 51

LOG (*plynx.constants.NodeResources* attribute), 50

LOG (*plynx.constants.resource_enums.NodeResources* attribute), 45

logger (in module *plynx.service.worker_server*), 77

logger (in module *plynx.web.common*), 84

logs (*plynx.db.node.CachedNode* attribute), 53

logs (*plynx.db.node.Node* attribute), 54

logs (*plynx.db.node_cache.NodeCache* attribute), 56

M

main() (in module *plynx.bin*), 41

make_enum() (in module *plynx.demo.basic_functions*), 61

make_fail_response() (in module *plynx.web*), 87

make_fail_response() (in module *plynx.web.common*), 84

make_int() (in module *plynx.demo.basic_functions*), 61

make_operations_meta() (in module *plynx.bin.cli*), 41

make_permission_denied() (in module *plynx.web.common*), 84

make_success_response() (in module *plynx.web.common*), 84

MANDATORY_DEPRECATED (*plynx.constants.node_enums.NodePostAction* attribute), 43

MANDATORY_DEPRECATED (*plynx.constants.NodePostAction* attribute), 48

MANDATORY_DEPRECATED (*plynx.constants.node_enums.NodeStatus* attribute), 43

MANDATORY_DEPRECATED (*plynx.constants.NodeStatus* attribute), 49

materialize_executor() (in module *plynx.utils.executor*), 80

materialize_fn_or_cls() (in module *plynx.plugins.executors.python.local*), 67

Meta (class in *plynx.db.db_object*), 52

min_count (*plynx.db.node.BaseResource* attribute), 53

min_count (*plynx.node.InputItem* attribute), 64

min_count (*plynx.node.OutputItem* attribute), 64

MISSING_INPUT (*plynx.constants.validation_enums.ValidationCode* attribute), 46

MISSING_INPUT (*plynx.constants.ValidationCode* attribute), 50

MISSING_PARAMETER (*plynx.constants.validation_enums.ValidationCode* attribute), 46

MISSING_PARAMETER (*plynx.constants.ValidationCode* attribute), 50

mode (*plynx.db.node.ParameterCode* attribute), 55

MODES (in module *plynx.service.cache*), 74

MODES (in module *plynx.service.users*), 75

MODIFY (*plynx.constants.UserPostAction* attribute), 50

MODIFY (*plynx.constants.users.UserPostAction* attribute), 46

MongoConfig (in module *plynx.utils.config*), 78

mutable_type (*plynx.db.node.Parameter* attribute), 55

N

name (*plynx.db.node.BaseResource* attribute), 53

name (*plynx.db.node.Parameter* attribute), 55

name (*plynx.node.InputItem* attribute), 64

name (*plynx.node.OutputItem* attribute), 64

name (*plynx.node.ParamItem* attribute), 64

nargs (*plynx.bin.cli.Arg* attribute), 41

Node (class in *plynx.db.node*), 53

NODE (*plynx.constants.validation_enums.ValidationTargetType* attribute), 46

NODE (*plynx.constants.ValidationTargetType* attribute), 51

NODE_CACHE (*plynx.constants.Collections* attribute), 47

NODE_CACHE (*plynx.constants.collections.Collections* attribute), 42

node_cache_manager (in module *plynx.service.cache*), 74

node_cache_manager() (in module *plynx.plugins.executors.dag*), 68

node_collection_manager (in module *plynx.web.run*), 86

node_collection_managers (in module *plynx.utils.node_utils*), 82

node_collection_managers (in module *plynx.web.node*), 85

node_id (*plynx.db.node.InputReference* attribute), 53

node_id (*plynx.db.node_cache.NodeCache* attribute), 56

node_inputs_and_params_are_identical() (in module *plynx.utils.node_utils*), 82

node_running_status (*plynx.base.executor.RunningStatus* attribute),

- 38
- node_running_status (*plynx.db.node.CachedNode attribute*), 53
- node_running_status (*plynx.db.node.Node attribute*), 54
- node_status (*plynx.db.node.Node attribute*), 54
- NODE_TO_NODE (*plynx.constants.node_enums.NodeClonePolicy attribute*), 44
- NODE_TO_NODE (*plynx.constants.NodeClonePolicy attribute*), 47
- NODE_TO_RUN (*plynx.constants.node_enums.NodeClonePolicy attribute*), 44
- NODE_TO_RUN (*plynx.constants.NodeClonePolicy attribute*), 47
- node_type (*plynx.node.PlynxParams attribute*), 64
- NodeCache (*class in plynx.db.node_cache*), 56
- NodeCacheManager (*class in plynx.db.node_cache_manager*), 56
- NodeClonePolicy (*class in plynx.constants*), 47
- NodeClonePolicy (*class in plynx.constants.node_enums*), 44
- NodeCollectionManager (*class in plynx.db.node_collection_manager*), 57
- NodeOrigin (*class in plynx.constants*), 47
- NodeOrigin (*class in plynx.constants.node_enums*), 44
- NodePostAction (*class in plynx.constants*), 47
- NodePostAction (*class in plynx.constants.node_enums*), 43
- NodePostStatus (*class in plynx.constants*), 48
- NodePostStatus (*class in plynx.constants.node_enums*), 43
- NodeResources (*class in plynx.constants*), 49
- NodeResources (*class in plynx.constants.resource_enums*), 45
- NodeRunningStatus (*class in plynx.constants*), 48
- NodeRunningStatus (*class in plynx.constants.node_enums*), 42
- NodeStatus (*class in plynx.constants*), 48
- NodeStatus (*class in plynx.constants.node_enums*), 43
- NodeVirtualCollection (*class in plynx.constants*), 49
- NodeVirtualCollection (*class in plynx.constants.node_enums*), 44
- O**
- object_id (*plynx.db.validation_error.ValidationError attribute*), 60
- offset (*plynx.base.hub.Query attribute*), 39
- open () (*in module plynx.utils.file_handler*), 81
- operation () (*in module plynx.node*), 65
- operation_manager (*in module plynx.web.node*), 85
- OperationConfig (*in module plynx.utils.config*), 78
- OPERATIONS (*plynx.constants.node_enums.NodeVirtualCollection attribute*), 44
- OPERATIONS (*plynx.constants.NodeVirtualCollection attribute*), 49
- origin (*plynx.db.node.Node attribute*), 54
- original_node_id (*plynx.db.node.Node attribute*), 54
- Output (*class in plynx.db.node*), 53
- OUTPUT (*plynx.constants.node_enums.SpecialNodeId attribute*), 44
- OUTPUT (*plynx.constants.NodeResources attribute*), 49
- OUTPUT (*plynx.constants.resource_enums.NodeResources attribute*), 45
- OUTPUT (*plynx.constants.SpecialNodeId attribute*), 49
- output () (*in module plynx.node*), 65
- OUTPUT_FILE_TYPE (*plynx.constants.HubSearchParams attribute*), 49
- OUTPUT_FILE_TYPE (*plynx.constants.resource_enums.HubSearchParams attribute*), 45
- output_id (*plynx.db.node.InputReference attribute*), 53
- OutputItem (*class in plynx.node*), 64
- OutputListTuple (*in module plynx.service.cache*), 74
- outputs (*plynx.db.node.CachedNode attribute*), 53
- outputs (*plynx.db.node.Node attribute*), 54
- outputs (*plynx.db.node_cache.NodeCache attribute*), 56
- outputs (*plynx.node.PlynxParams attribute*), 64
- P**
- PAGINATION_QUERY_KEYS (*in module plynx.web.node*), 85
- PARAM (*plynx.constants.NodeResources attribute*), 49
- PARAM (*plynx.constants.resource_enums.NodeResources attribute*), 45
- param () (*in module plynx.node*), 65
- Parameter (*class in plynx.db.node*), 55
- parameter (*in module plynx.node*), 65
- PARAMETER (*plynx.constants.validation_enums.ValidationTargetType attribute*), 46
- PARAMETER (*plynx.constants.ValidationTargetType attribute*), 51
- parameter_type (*plynx.db.node.Parameter attribute*), 55
- parameter_type (*plynx.node.ParamItem attribute*), 64
- ParameterCode (*class in plynx.db.node*), 55
- ParameterEnum (*class in plynx.db.node*), 55
- ParameterListOfNodes (*class in plynx.db.node*), 55
- parameters (*plynx.db.node.Node attribute*), 54
- ParameterTypes (*class in plynx.constants*), 49

ParameterTypes (class in `pl-
ynx.constants.parameter_types`), 44
ParamItem (class in `pl-
ynx.node`), 64
params (`pl-
ynx.node.PlynxParams` attribute), 64
parent_node_id (`pl-
ynx.db.node.Node` attribute), 54
parse_global_config_parameters() (pl-
ynx.bin.cli.CLIFactory class method), 41
parse_global_config_parameters() (pl-
ynx.bin.CLIFactory class method), 41
parse_search_string() (in module `pl-
ynx.utils.common`), 77
password_hash (`pl-
ynx.db.user.User` attribute), 59
PDF (class in `pl-
ynx.plugins.resources.common`), 73
per_page (`pl-
ynx.base.hub.Query` attribute), 39
pick_a_run() (in module `pl-
ynx.web.run`), 86
pick_node() (`pl-
ynx.db.node_collection_manager.NodeCollectionManager` method), 58
picture (`pl-
ynx.db.user.UserSettings` attribute), 59
PLUGINS_DICT (in module `pl-
ynx.web.node`), 85
PLUGINS_DICT (in module `pl-
ynx.web.state`), 86
PluginsConfig (in module `pl-
ynx.utils.config`), 78
plynx (module), 37
plynx.base (module), 37
plynx.base.executor (module), 38
plynx.base.hub (module), 39
plynx.base.resource (module), 39
plynx.bin (module), 40
plynx.bin.cli (module), 40
plynx.constants (module), 42
plynx.constants.collections (module), 42
plynx.constants.node_enums (module), 42
plynx.constants.parameter_types (module), 44
plynx.constants.resource_enums (module), 45
plynx.constants.users (module), 45
plynx.constants.validation_enums (mod-
ule), 46
plynx.constants.web (module), 47
plynx.db (module), 51
plynx.db.db_object (module), 51
plynx.db.demo_user_manager (module), 52
plynx.db.node (module), 52
plynx.db.node_cache (module), 56
plynx.db.node_cache_manager (module), 56
plynx.db.node_collection_manager (mod-
ule), 57
plynx.db.run_cancellation_manager (mod-
ule), 58
plynx.db.user (module), 59
plynx.db.validation_error (module), 60
plynx.db.worker_state (module), 61
plynx.demo (module), 61
plynx.demo.basic_functions (module), 61
pl-
ynx.demo.hello_world (module), 62
pl-
ynx.demo.types (module), 62
pl-
ynx.node (module), 63
pl-
ynx.node.typing (module), 63
pl-
ynx.node.utils (module), 63
pl-
ynx.plugins (module), 65
pl-
ynx.plugins.executors (module), 65
pl-
ynx.plugins.executors.bases (module), 67
pl-
ynx.plugins.executors.dag (module), 68
pl-
ynx.plugins.executors.local (module), 69
pl-
ynx.plugins.executors.python (module), 65
pl-
ynx.plugins.executors.python.dag (mod-
ule), 65
pl-
ynx.plugins.executors.python.local
module, 66
pl-
ynx.plugins.hubs (module), 70
pl-
ynx.plugins.hubs.collection (module), 70
pl-
ynx.plugins.hubs.static_list (module), 71
pl-
ynx.plugins.resources (module), 71
pl-
ynx.plugins.resources.cloud_resources
(module), 72
pl-
ynx.plugins.resources.common (module), 72
pl-
ynx.plugins.resources.python (module), 71
pl-
ynx.plugins.resources.python.common
(module), 71
pl-
ynx.service (module), 74
pl-
ynx.service.cache (module), 74
pl-
ynx.service.execute (module), 75
pl-
ynx.service.make_operations_meta (mod-
ule), 75
pl-
ynx.service.users (module), 75
pl-
ynx.service.worker (module), 76
pl-
ynx.service.worker_server (module), 76
pl-
ynx.utils (module), 77
pl-
ynx.utils.common (module), 77
pl-
ynx.utils.config (module), 78
pl-
ynx.utils.content (module), 79
pl-
ynx.utils.db_connector (module), 79
pl-
ynx.utils.exceptions (module), 80
pl-
ynx.utils.executor (module), 80
pl-
ynx.utils.file_handler (module), 81
pl-
ynx.utils.hub_node_registry (module), 81
pl-
ynx.utils.logs (module), 81
pl-
ynx.utils.node_utils (module), 82
pl-
ynx.utils.plugin_manager (module), 83
pl-
ynx.utils.thumbnails (module), 83
pl-
ynx.web (module), 84
pl-
ynx.web.common (module), 84
pl-
ynx.web.health (module), 85
pl-
ynx.web.node (module), 85

- plynx.web.resource (module), 85
 plynx.web.run (module), 86
 plynx.web.state (module), 86
 plynx.web.user (module), 86
 PLYNX_CONFIG_PATH (in module plynx.utils.config), 78
 PLYNX_DB (in module plynx.utils.db_connector), 79
 PLynxAsyncExecutor (class in plynx.plugins.executors.bases), 67
 PLynxAsyncExecutorWithDirectory (class in plynx.plugins.executors.bases), 68
 PLynxParams (class in plynx.node), 64
 PLynxSyncExecutor (class in plynx.plugins.executors.bases), 68
 policies (plynx.db.user.User attribute), 59
 POOL_SIZE (in module plynx.plugins.executors.python.dag), 65
 pop_jobs () (plynx.plugins.executors.dag.DAG method), 68
 pop_jobs () (plynx.plugins.executors.python.dag.DAGParallel method), 65
 post () (plynx.db.node_cache_manager.NodeCacheManager static method), 57
 post_node () (in module plynx.web.node), 85
 post_register () (in module plynx.web.user), 87
 post_register_with_oauth2 () (in module plynx.web.user), 87
 post_request () (in module plynx.utils.executor), 80
 post_resource () (in module plynx.web.resource), 86
 post_search_nodes () (in module plynx.web.node), 85
 post_user () (in module plynx.web.user), 87
 postprocess_output () (plynx.base.resource.BaseResource static method), 40
 postprocess_output () (plynx.plugins.resources.common.Directory static method), 74
 postprocess_output () (plynx.plugins.resources.python.common.Json static method), 71
 prep_args () (in module plynx.plugins.executors.python.local), 67
 prepare_input () (plynx.base.resource.BaseResource static method), 40
 prepare_input () (plynx.plugins.resources.cloud_resources.CloudStorage static method), 72
 prepare_input () (plynx.plugins.resources.common.Directory static method), 74
 prepare_input () (plynx.plugins.resources.common.Executable static method), 74
 prepare_output () (plynx.base.resource.BaseResource static method), 40
 prepare_output () (plynx.plugins.resources.cloud_resources.CloudStorage static method), 72
 prepare_output () (plynx.plugins.resources.common.Directory static method), 74
 prepare_parameters_for_python () (in module plynx.plugins.executors.local), 69
 preprocess_input () (plynx.base.resource.BaseResource static method), 40
 preprocess_input () (plynx.plugins.resources.common.RawColor static method), 73
 preprocess_input () (plynx.plugins.resources.common.RawFloat static method), 73
 preprocess_input () (plynx.plugins.resources.common.RawInt static method), 72
 preprocess_input () (plynx.plugins.resources.python.common.Json static method), 71
 preview () (plynx.base.resource.BaseResource class method), 40
 preview () (plynx.plugins.resources.cloud_resources.CloudStorage class method), 72
 preview () (plynx.plugins.resources.common._BaseSeparated class method), 73
 preview () (plynx.plugins.resources.common.Directory class method), 74
 preview () (plynx.plugins.resources.common.Image class method), 73
 preview () (plynx.plugins.resources.common.Json class method), 74
 preview () (plynx.plugins.resources.common.PDF class method), 73
 preview () (plynx.plugins.resources.python.common.Json class method), 72
 PREVIEW_CMD (plynx.constants.node_enums.NodePostAction attribute), 43
 PREVIEW_CMD (plynx.constants.NodePostAction attribute), 48
 PreviewObject (in module plynx.base.resource), 40
 primitive_override (plynx.db.node.Input attribute), 53
 PRIMITIVE_TYPES (in module plynx.constants), 49
 PRIMITIVE_TYPES (in module plynx.constants.parameter_types), 45

print_any() (in module *plynx.demo.types*), 62
 print_bool() (in module *plynx.demo.types*), 62
 print_color() (in module *plynx.demo.types*), 62
 print_dict() (in module *plynx.demo.types*), 62
 print_float() (in module *plynx.demo.types*), 62
 print_int() (in module *plynx.demo.types*), 62
 print_message() (in module *plynx.demo.hello_world*), 62
 print_str() (in module *plynx.demo.types*), 62
 PROPERTY (*plynx.constants.validation_enums.ValidationTargetType* attribute), 46
 PROPERTY (*plynx.constants.ValidationTargetType* attribute), 51
 protected (*plynx.db.node_cache.NodeCache* attribute), 56
 publicable (*plynx.db.node.Parameter* attribute), 55
 PUBSUB (*plynx.service.worker_server.RunEnv* attribute), 77
 push_worker_state() (in module *plynx.web.state*), 86
 PythonNode (class in *plynx.plugins.executors.local*), 70
 PythonNode (class in *plynx.plugins.executors.python.local*), 67

Q

Query (class in *plynx.base.hub*), 39
 query_yes_no() (in module *plynx.utils.common*), 77

R

Raw (class in *plynx.plugins.resources.common*), 72
 RawColor (class in *plynx.plugins.resources.common*), 73
 RawFloat (class in *plynx.plugins.resources.common*), 72
 RawInt (class in *plynx.plugins.resources.common*), 72
 READY (*plynx.constants.node_enums.NodeRunningStatus* attribute), 42
 READY (*plynx.constants.node_enums.NodeStatus* attribute), 43
 READY (*plynx.constants.NodeRunningStatus* attribute), 48
 READY (*plynx.constants.NodeStatus* attribute), 49
 REARRANGE_NODES (*plynx.constants.node_enums.NodePostAction* attribute), 43
 REARRANGE_NODES (*plynx.constants.NodePostAction* attribute), 48
 redirect_to_plynx_logs (class in *plynx.plugins.executors.python.local*), 67
 reference (*plynx.db.node.Parameter* attribute), 55
 REFRESH_TOKEN (*plynx.constants.TokenType* attribute), 50

REFRESH_TOKEN (*plynx.constants.users.TokenType* attribute), 46
 register_class() (in module *plynx.db.db_object*), 51
 register_list_item() (in module *plynx.plugins.hubs.static_list*), 71
 register_node() (*plynx.utils.hub_node_registry.Registry* method), 81
 register_user() (in module *plynx.web.common*), 84
 RegisterUserException, 80
 RegisterUserExceptionCode (class in *plynx.constants*), 50
 RegisterUserExceptionCode (class in *plynx.constants.users*), 46
 Registry (class in *plynx.utils.hub_node_registry*), 81
 registry (in module *plynx.utils.hub_node_registry*), 81
 removable (*plynx.db.node.Parameter* attribute), 55
 remove() (*plynx.db.run_cancellation_manager.RunCancellationManager* static method), 59
 remove_auto_run_disabled() (in module *plynx.utils.node_utils*), 82
 removed (*plynx.db.node_cache.NodeCache* attribute), 56
 request_task() (*plynx.plugins.executors.python.dag.ExecutorWithWebWorkerServer* static method), 66
 REQUESTS_TIMEOUT (in module *plynx.utils.executor*), 80
 required (*plynx.bin.cli.Arg* attribute), 40
 requires_auth() (in module *plynx.web*), 87
 requires_auth() (in module *plynx.web.common*), 84
 reset_nodes() (in module *plynx.utils.node_utils*), 82
 RESOURCE_TYPES (in module *plynx.web.resource*), 86
 ResourceConfig (in module *plynx.utils.config*), 78
 ResponseStatus (class in *plynx.constants*), 51
 ResponseStatus (class in *plynx.constants.web*), 47
 RESTORED (*plynx.constants.node_enums.NodeRunningStatus* attribute), 42
 RESTORED (*plynx.constants.NodeRunningStatus* attribute), 48
 run() (*plynx.base.executor.BaseExecutor* method), 38
 run() (*plynx.base.executor.Dummy* method), 39
 run() (*plynx.plugins.executors.dag.DAG* method), 69
 run() (*plynx.plugins.executors.local.BaseBash* method), 70
 run() (*plynx.plugins.executors.local.BashJinja2* method), 70
 run() (*plynx.plugins.executors.local.File* method), 70
 run() (*plynx.plugins.executors.local.PythonNode* method), 70

- `run()` (*plynx.plugins.executors.python.dag.DAG method*), 66
- `run()` (*plynx.plugins.executors.python.local.PythonNode method*), 67
- `run()` (*plynx.utils.executor.DBJobExecutor method*), 81
- `run_api()` (*in module plynx.web*), 87
- `run_api()` (*in module plynx.web.common*), 85
- `run_cache()` (*in module plynx.service.cache*), 75
- `run_cancellation_manager` (*in module plynx.web.node*), 85
- `run_cancellation_manager` (*in module plynx.web.run*), 86
- `run_cancellation_manager()` (*in module plynx.plugins.executors.bases*), 67
- `RUN_CANCELLATIONS` (*plynx.constants.Collections attribute*), 47
- `RUN_CANCELLATIONS` (*plynx.constants.collections.Collections attribute*), 42
- `run_clean_cache()` (*in module plynx.service.cache*), 74
- `run_create_user()` (*in module plynx.service.users*), 75
- `run_execute()` (*in module plynx.service.execute*), 75
- `run_id` (*plynx.db.node_cache.NodeCache attribute*), 56
- `run_id` (*plynx.db.run_cancellation_manager.RunCancellation attribute*), 58
- `run_id` (*plynx.db.run_cancellation_manager.RunCancellationManager attribute*), 58
- `run_list_cache()` (*in module plynx.service.cache*), 74
- `run_list_users()` (*in module plynx.service.users*), 75
- `run_make_operations_meta()` (*in module plynx.service.make_operations_meta*), 75
- `run_set_activation()` (*in module plynx.service.users*), 75
- `RUN_TO_NODE` (*plynx.constants.node_enums.NodeClonePolicy attribute*), 44
- `RUN_TO_NODE` (*plynx.constants.NodeClonePolicy attribute*), 47
- `run_users()` (*in module plynx.service.users*), 75
- `run_worker()` (*in module plynx.service.worker*), 76
- `run_worker_server()` (*in module plynx.service.worker_server*), 77
- `RunCancellation` (*class in plynx.db.run_cancellation_manager*), 58
- `RunCancellationManager` (*class in plynx.db.run_cancellation_manager*), 58
- `RunEnv` (*class in plynx.service.worker_server*), 77
- `RUNNING` (*plynx.constants.node_enums.NodeRunningStatus attribute*), 42
- `RUNNING` (*plynx.constants.NodeRunningStatus attribute*), 48
- `RunningStatus` (*class in plynx.base.executor*), 38
- `RUNS` (*plynx.constants.Collections attribute*), 47
- `RUNS` (*plynx.constants.collections.Collections attribute*), 42
- `runs` (*plynx.db.worker_state.WorkerState attribute*), 61
- ## S
- `SAVE` (*plynx.constants.node_enums.NodePostAction attribute*), 43
- `SAVE` (*plynx.constants.NodePostAction attribute*), 47
- `save()` (*plynx.db.db_object.DBObject method*), 51
- `SDB_STATUS_UPDATE_TIMEOUT` (*plynx.service.worker.Worker attribute*), 76
- `search` (*plynx.base.hub.Query attribute*), 39
- `search()` (*plynx.base.hub.BaseHub method*), 39
- `search()` (*plynx.plugins.hubs.collection.CollectionHub method*), 71
- `search()` (*plynx.plugins.hubs.static_list.StaticListHub method*), 71
- `SEARCH_RGX` (*in module plynx.utils.common*), 77
- `SearchParameter` (*in module plynx.utils.common*), 77
- `SEPARATOR` (*plynx.plugins.resources.common._BaseSeparated attribute*), 73
- `SEPARATOR` (*plynx.plugins.resources.common.CSV attribute*), 73
- `SEPARATOR` (*plynx.plugins.resources.common.TSV attribute*), 73
- `serve_forever()` (*plynx.service.worker.Worker method*), 76
- `set_logging_level()` (*in module plynx.utils.logs*), 82
- `set_parameter()` (*in module plynx.utils.config*), 79
- `settings` (*plynx.db.user.User attribute*), 59
- `sleep()` (*in module plynx.demo.basic_functions*), 61
- `SPECIAL` (*plynx.constants.node_enums.NodeRunningStatus attribute*), 42
- `SPECIAL` (*plynx.constants.NodeRunningStatus attribute*), 48
- `SpecialNodeId` (*class in plynx.constants*), 49
- `SpecialNodeId` (*class in plynx.constants.node_enums*), 44
- `starred` (*plynx.db.node.Node attribute*), 54
- `stateful_class_registry` (*in module plynx.plugins.executors.python.local*), 67
- `stateful_init_mutex` (*in module plynx.plugins.executors.python.local*), 66
- `Statefull` (*class in plynx.demo.basic_functions*), 61
- `STATIC` (*plynx.constants.node_enums.NodeRunningStatus attribute*), 42
- `STATIC` (*plynx.constants.NodeRunningStatus attribute*), 48
- `StaticListHub` (*class in plynx.plugins.hubs.static_list*), 71

status (*plynx.base.hub.Query attribute*), 39
 status () (*plynx.base.executor.Dummy method*), 39
 stop () (*plynx.service.worker.Worker method*), 76
 StorageConfig (*in module plynx.utils.config*), 78
 STR (*plynx.constants.parameter_types.ParameterTypes attribute*), 44
 STR (*plynx.constants.ParameterTypes attribute*), 49
 str_to_bool () (*in module plynx.utils.common*), 77
 STR_TO_CLASS (*in module plynx.node.typing*), 63
 SUBPARSERS (*plynx.bin.cli.CLIFactory attribute*), 41
 SUBPARSERS (*plynx.bin.CLIFactory attribute*), 41
 SUCCESS (*plynx.constants.node_enums.NodePostStatus attribute*), 43
 SUCCESS (*plynx.constants.node_enums.NodeRunningStatus attribute*), 42
 SUCCESS (*plynx.constants.NodePostStatus attribute*), 48
 SUCCESS (*plynx.constants.NodeRunningStatus attribute*), 48
 SUCCESS (*plynx.constants.ResponseStatus attribute*), 51
 SUCCESS (*plynx.constants.web.ResponseStatus attribute*), 47
 successor_node_id (*plynx.db.node.Node attribute*), 54

T

target (*plynx.db.validation_error.ValidationError attribute*), 60
 template_collection_manager (*in module plynx.db.demo_user_manager*), 52
 template_collection_manager (*in module plynx.web.user*), 87
 template_node_id (*plynx.db.node.Node attribute*), 54
 TEMPLATES (*plynx.constants.Collections attribute*), 47
 TEMPLATES (*plynx.constants.collections.Collections attribute*), 42
 TEXT (*plynx.constants.parameter_types.ParameterTypes attribute*), 44
 TEXT (*plynx.constants.ParameterTypes attribute*), 49
 thumbnail (*plynx.db.node.Output attribute*), 53
 thumbnail () (*plynx.base.resource.BaseResource class method*), 40
 thumbnail () (*plynx.plugins.resources.common.Image class method*), 73
 thumbnail () (*plynx.plugins.resources.python.common.Json class method*), 72
 TICK_TIMEOUT (*plynx.utils.executor.TickThread attribute*), 80
 TickThread (*class in plynx.utils.executor*), 80
 title (*plynx.db.node.Node attribute*), 54
 title (*plynx.node.PlynxParams attribute*), 64
 title (*plynx.node.utils.Group attribute*), 63
 to_dict () (*plynx.db.db_object.DBObject method*), 52

to_dict () (*plynx.node.InputItem method*), 64
 to_dict () (*plynx.node.OutputItem method*), 64
 to_dict () (*plynx.node.ParamItem method*), 64
 to_dict () (*plynx.node.utils.Group method*), 63
 to_object_id () (*in module plynx.utils.common*), 77
 TokenType (*class in plynx.constants*), 50
 TokenType (*class in plynx.constants.users*), 46
 traverse_in_order () (*in module plynx.utils.node_utils*), 82
 traverse_left_join () (*in module plynx.utils.node_utils*), 83
 traverse_reversed () (*in module plynx.utils.node_utils*), 82
 TRUES (*in module plynx.utils.common*), 77
 TSV (*class in plynx.plugins.resources.common*), 73
 type (*plynx.bin.cli.Arg attribute*), 40
 type_to_str () (*in module plynx.node.typing*), 63
 types_group (*in module plynx.demo*), 63

U

update_dict_recursively () (*in module plynx.utils.common*), 78
 update_node () (*plynx.plugins.executors.dag.DAG method*), 68
 update_run () (*in module plynx.web.run*), 86
 UPGRADE_NODES (*plynx.constants.node_enums.NodePostAction attribute*), 43
 UPGRADE_NODES (*plynx.constants.NodePostAction attribute*), 48
 upgrade_sub_nodes () (*plynx.db.node_collection_manager.NodeCollectionManager method*), 58
 upload_file () (*in module plynx.web.resource*), 86
 upload_file_stream () (*in module plynx.utils.file_handler*), 81
 upload_logs () (*plynx.plugins.executors.local.BaseBash method*), 70
 urljoin () (*in module plynx.utils.executor*), 80
 User (*class in plynx.db.user*), 59
 user_id (*plynx.base.hub.Query attribute*), 39
 UserCollectionManager (*class in plynx.db.user*), 60
 username (*plynx.db.user.User attribute*), 59
 USERNAME_ALREADY_EXISTS (*plynx.constants.RegisterUserExceptionCode attribute*), 50
 USERNAME_ALREADY_EXISTS (*plynx.constants.users.RegisterUserExceptionCode attribute*), 46
 UserPostAction (*class in plynx.constants*), 50
 UserPostAction (*class in plynx.constants.users*), 45
 USERS (*plynx.constants.Collections attribute*), 47

- USERS (*plynx.constants.collections.Collections attribute*), 42
- users() (*in module plynx.bin.cli*), 41
- UserSettings (*class in plynx.db.user*), 59
- ## V
- VALIDATE (*plynx.constants.node_enums.NodePostAction attribute*), 43
- VALIDATE (*plynx.constants.NodePostAction attribute*), 47
- validate() (*plynx.base.executor.BaseExecutor method*), 39
- validate() (*plynx.plugins.executors.dag.DAG method*), 69
- validation_code (*plynx.db.validation_error.ValidationError attribute*), 60
- VALIDATION_FAILED (*plynx.constants.node_enums.NodePostStatus attribute*), 44
- VALIDATION_FAILED (*plynx.constants.NodePostStatus attribute*), 48
- ValidationCode (*class in plynx.constants*), 50
- ValidationCode (*class in plynx.constants.validation_enums*), 46
- ValidationError (*class in plynx.db.validation_error*), 60
- ValidationTargetType (*class in plynx.constants*), 50
- ValidationTargetType (*class in plynx.constants.validation_enums*), 46
- value (*plynx.db.node.Parameter attribute*), 55
- value (*plynx.db.node.ParameterCode attribute*), 55
- value (*plynx.db.node.ParameterListOfNodes attribute*), 55
- value (*plynx.node.ParamItem attribute*), 64
- values (*plynx.db.node._BaseResource attribute*), 53
- values (*plynx.db.node.ParameterEnum attribute*), 55
- verify_auth_token() (*plynx.db.user.User static method*), 60
- verify_password() (*in module plynx.web*), 87
- verify_password() (*in module plynx.web.common*), 84
- verify_password() (*plynx.db.user.User method*), 59
- version() (*in module plynx.bin.cli*), 41
- ## W
- WEB_CONFIG (*in module plynx.plugins.resources.common*), 72
- WebConfig (*in module plynx.utils.config*), 78
- widget (*plynx.db.node.Parameter attribute*), 55
- widget (*plynx.node.ParamItem attribute*), 64
- Worker (*class in plynx.service.worker*), 76
- worker() (*in module plynx.bin.cli*), 41
- worker_id (*plynx.db.worker_state.WorkerState attribute*), 61
- worker_main() (*in module plynx.plugins.executors.python.dag*), 65
- worker_server() (*in module plynx.bin.cli*), 41
- WORKER_STATE (*plynx.constants.Collections attribute*), 47
- WORKER_STATE (*plynx.constants.collections.Collections attribute*), 42
- WORKER_STATE_UPDATE_TIMEOUT (*plynx.service.worker.Worker attribute*), 76
- worker_states() (*in module plynx.web.state*), 86
- WorkerConfig (*in module plynx.utils.config*), 78
- WorkerState (*class in plynx.db.worker_state*), 61
- workflow_manager (*in module plynx.utils.content*), 79
- workflow_manager (*in module plynx.web.node*), 85
- WorkflowConfig (*in module plynx.utils.config*), 78
- WORKFLOWS (*plynx.constants.node_enums.NodeVirtualCollection attribute*), 44
- WORKFLOWS (*plynx.constants.NodeVirtualCollection attribute*), 49
- ## X
- x (*plynx.db.node.Node attribute*), 54
- ## Y
- y (*plynx.db.node.Node attribute*), 54
- ## Z
- zipdir() (*in module plynx.utils.common*), 77