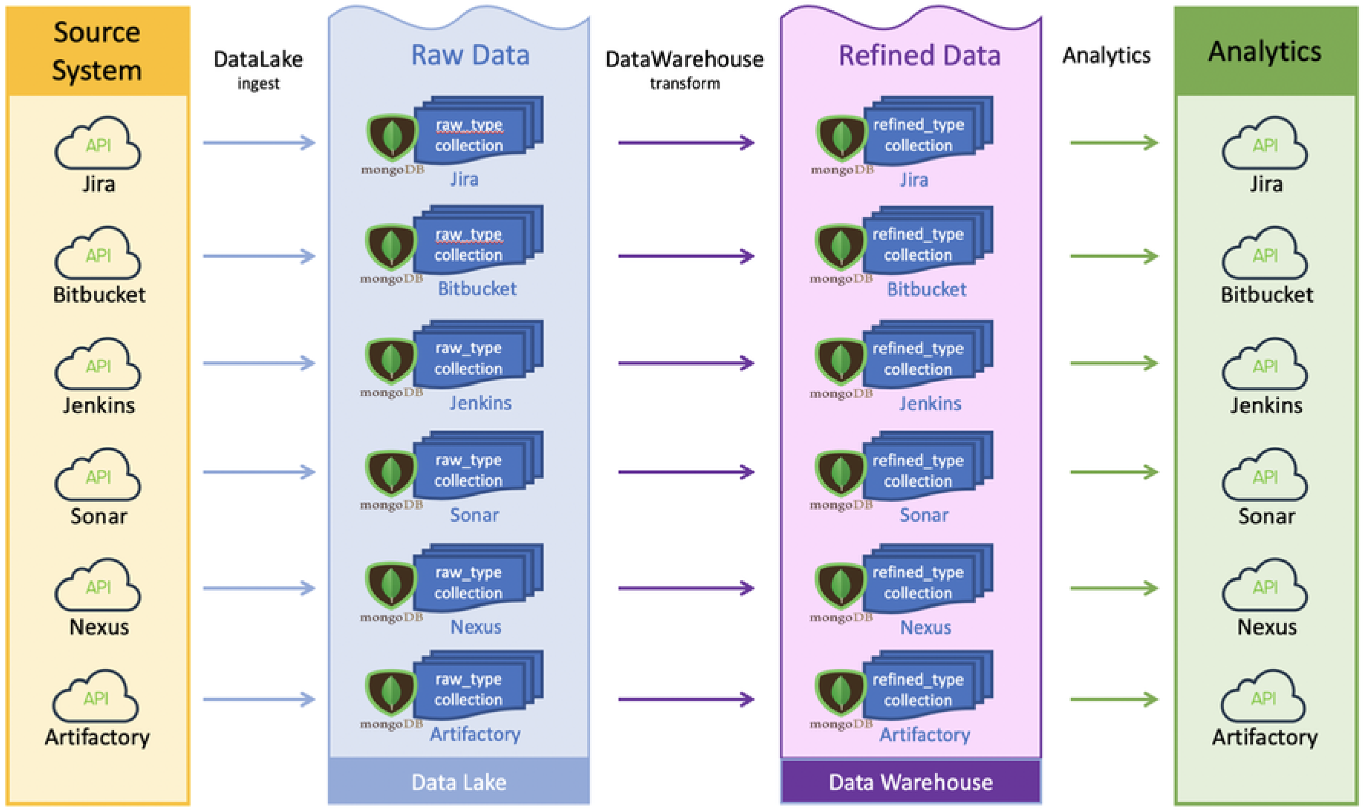
DevOps Platform - Analytics

High Level Architecture



900

Analytics Details

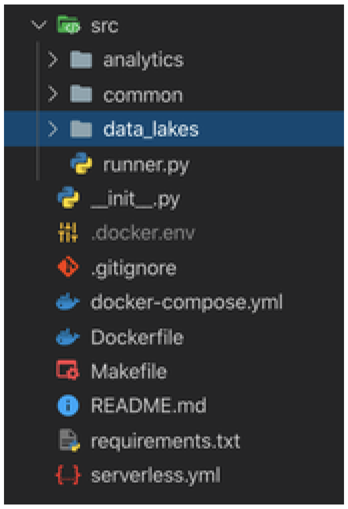
Jira Analytics Details...

Contents from DevOps Platform - Jira Analytics

Overview

The DevOps Platform - Jira Analytics offering provides metrics and machine learning capabilities derived from the data pulled from the Jira API.

Code Structure



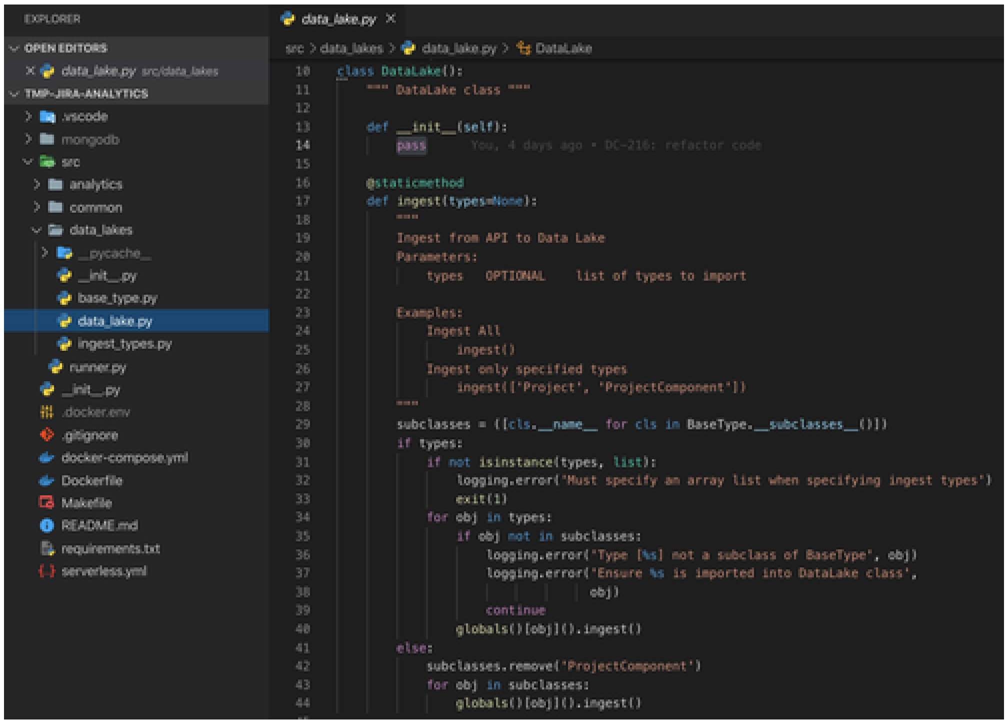
```
src
  analytics
  common
  data_lakes
  runner.py
__init__.py
.docker.env
.gitignore
docker-compose.yml
Dockerfile
Makefile
README.md
requirements.txt
serverless.yml
```

Path	Description
src/analytics	Code that performs analytics against the raw data stored in the Data Lakes (mongodb). **Note:** Analytics should be performed against the database and **NOT** against the Jira API directly.
src/common	Common helper functions
src/data_lakes	Code responsible for the data ingestion process.

Data Ingestion

The Data Ingestion process pulls data from the Jira API and pushes it into the Data Lake as-is into the <type>_raw collections without any data cleansing or transformation.

The ingest process starts from the src/data_lakes/data_lake.py DataLake().ingest(types=None).



```
EXPLORER
OPEN EDITORS
  × data_lake.py src/data_lakes
TMP-JIRA-ANALYTICS
  .vscode
  mongodb
  src
    analytics
    common
    data_lakes
      __pycache__
      __init__.py
      base_type.py
      data_lake.py
      ingest_types.py
    runner.py
  __init__.py
  .docker.env
  .gitignore
  docker-compose.yml
  Dockerfile
  Makefile
  README.md
  requirements.txt
  serverless.yml
```

```python
class DataLake():
    """ DataLake class """

    def __init__(self):
        pass        You, 4 days ago • DC-216: refactor code

    @staticmethod
    def ingest(types=None):
        """
        Ingest from API to Data Lake
        Parameters:
            types   OPTIONAL    list of types to import

        Examples:
            Ingest All
                ingest()
            Ingest only specified types
                ingest(['Project', 'ProjectComponent'])
        """
        subclasses = ([cls.__name__ for cls in BaseType.__subclasses__()])
        if types:
            if not isinstance(types, list):
                logging.error('Must specify an array list when specifying ingest types')
                exit(1)
            for obj in types:
                if obj not in subclasses:
                    logging.error('Type [%s] not a subclass of BaseType', obj)
                    logging.error('Ensure %s is imported into DataLake class',
                                  obj)
                    continue
                globals()[obj]().ingest()
        else:
            subclasses.remove('ProjectComponent')
            for obj in subclasses:
                globals()[obj]().ingest()
```

If no types are defined, it will ingest data from all Jira API endpoints using the classes defined in src/data_lakes/ingest_types.py.

The method also allows the option to only ingest the types provided in the parameter list.

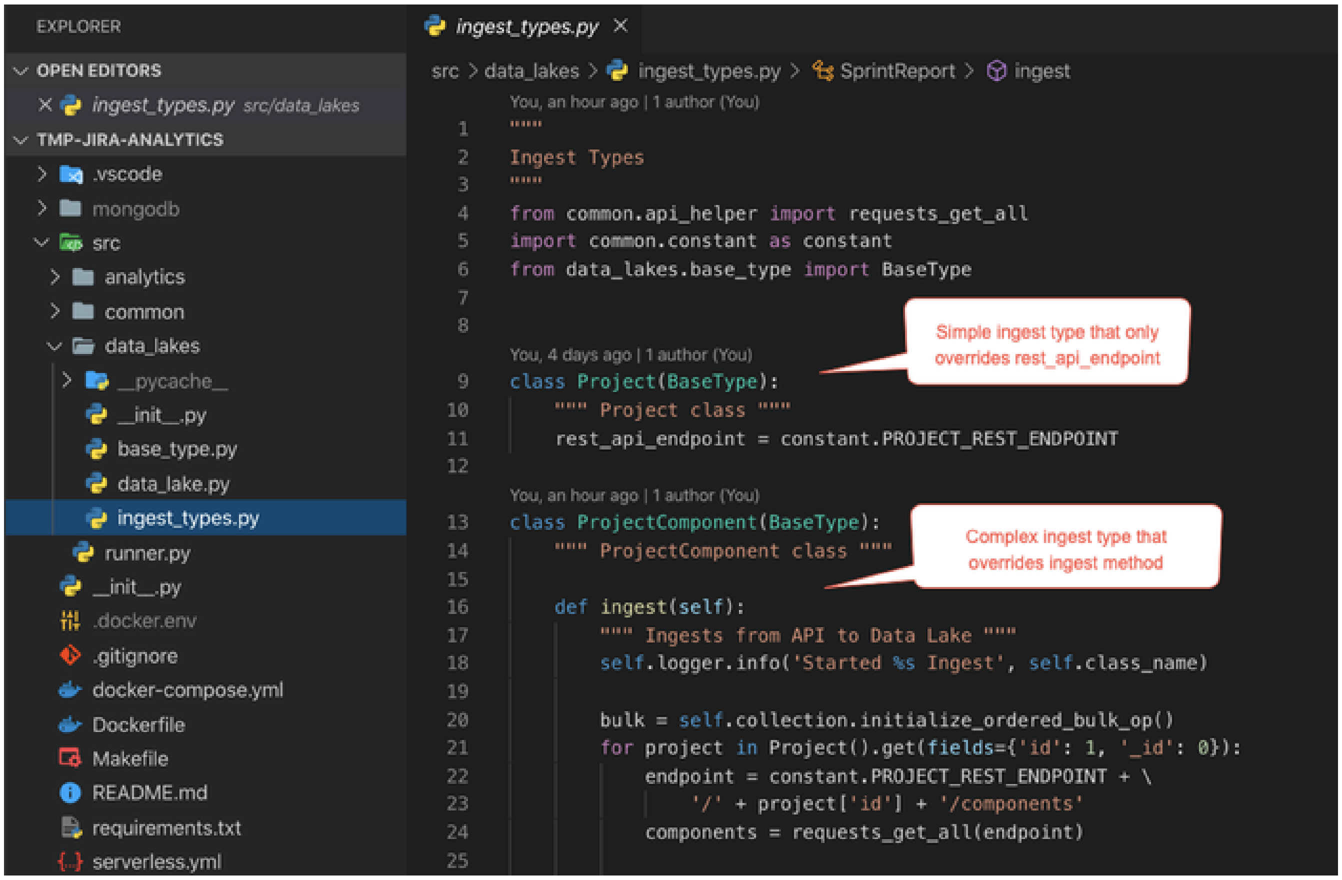
```
# Example of ingesting all types
DataLake().ingest()

# Example of ingesting a few types
DataLake().ingest(['Project','Priority'])
```

Ingest Class Definitions

The ingest classes defined in src/data_lakes/ingest_types.py are subclasses from the base class src/data_lakes/base_type.py.

For most ingest classes, only the rest_api_endpoint needs to be overridden.

Complex types may require an override of the ingest method.


EXPLORER
OPEN EDITORS
ingest_types.py src/data_lakes
TMP-JIRA-ANALYTICS
.vscode
mongodb
src
analytics
common
data_lakes
__pycache__
__init__.py
base_type.py
data_lake.py
ingest_types.py
runner.py
__init__.py
.docker.env
.gitignore
docker-compose.yml
Dockerfile
Makefile
README.md
requirements.txt
serverless.yml
ingest_types.py
src > data_lakes > ingest_types.py > SprintReport > ingest
You, an hour ago | 1 author (You)
You, 4 days ago | 1 author (You)
You, an hour ago | 1 author (You)
Simple ingest type that only overrides rest_api_endpoint
Complex ingest type that overrides ingest method

```
"""
Ingest Types
"""
from common.api_helper import requests_get_all
import common.constant as constant
from data_lakes.base_type import BaseType


class Project(BaseType):
    """ Project class """
    rest_api_endpoint = constant.PROJECT_REST_ENDPOINT

class ProjectComponent(BaseType):
    """ ProjectComponent class """

    def ingest(self):
        """ Ingests from API to Data Lake """
        self.logger.info('Started %s Ingest', self.class_name)

        bulk = self.collection.initialize_ordered_bulk_op()
        for project in Project().get(fields={'id': 1, '_id': 0}):
            endpoint = constant.PROJECT_REST_ENDPOINT + \
                '/' + project['id'] + '/components'
            components = requests_get_all(endpoint)

```

Ingest Frequency

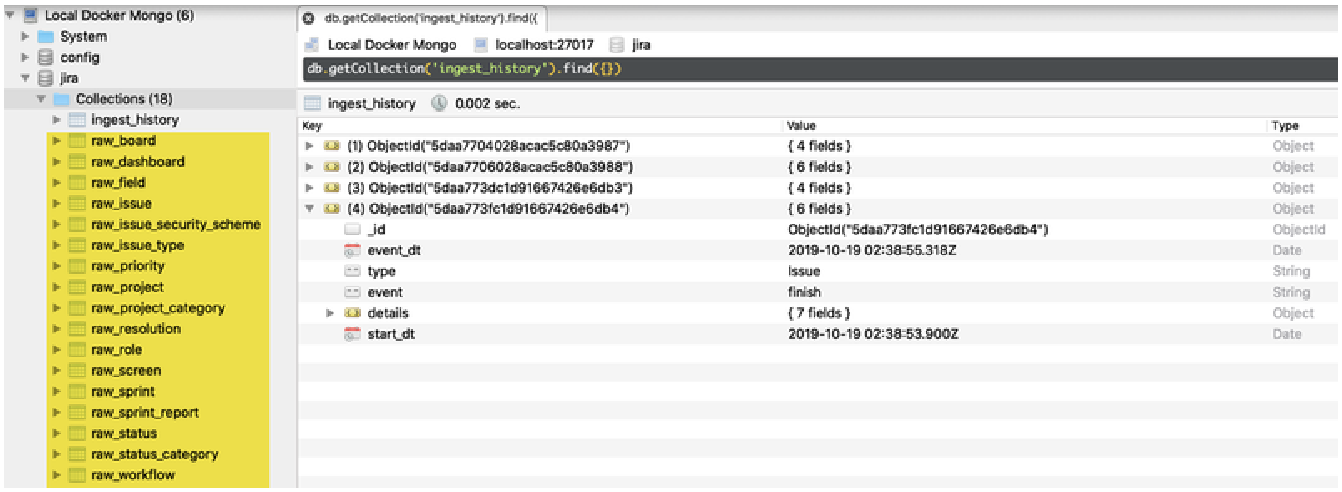
The decision on how frequent a full ingest should be executed will be determined on a case-by-case basis. In general, the plan is to run a full ingest at least once a day.

Certain endpoints that have a large amount of items (like Jira Issues) will be fined tuned to track when the last ingest occurred and only process data since the last ingest.

Other types may have additional flags to help determine if their child types need to be updated as part of the ingest. Example: sprint type may have a flag to not update the existing associated sprint report if the sprint state=closed.

Data Lake - Collections

Each Ingest type will store the data pulled from the Jira API to a collection called raw_<type>.

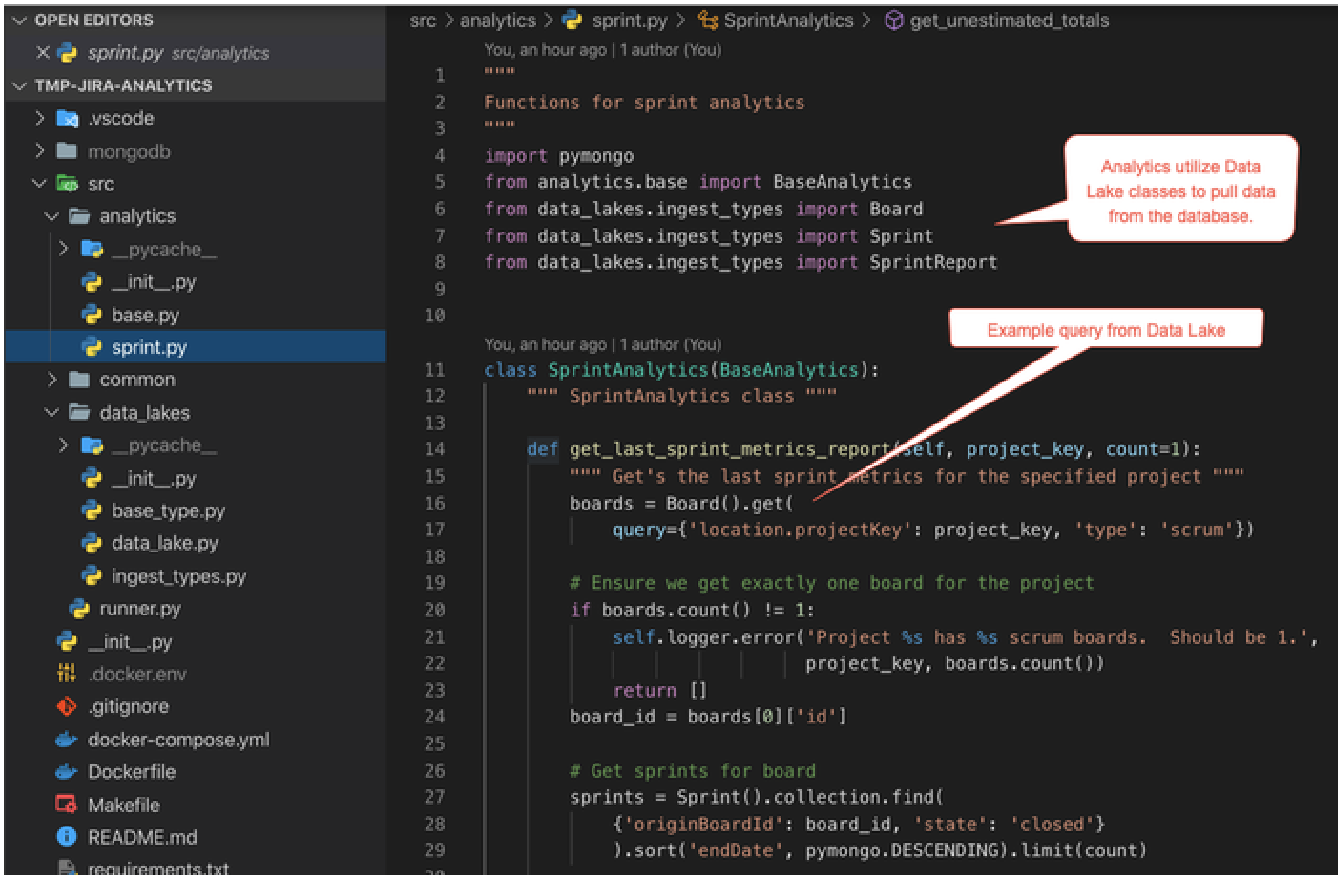


```
Local Docker Mongo (6)
  System
  config
  jira
    Collections (18)
      ingest_history
      raw_board
      raw_dashboard
      raw_field
      raw_issue
      raw_issue_security_scheme
      raw_issue_type
      raw_priority
      raw_project
      raw_project_category
      raw_resolution
      raw_role
      raw_screen
      raw_sprint
      raw_sprint_report
      raw_status
      raw_status_category
      raw_workflow
```

```
db.getCollection('ingest_history').find({
Local Docker Mongo   localhost:27017   jira
db.getCollection('ingest_history').find({})

ingest_history   0.002 sec.
Key                                             Value                                      Type
(1) ObjectId("5daa7704028acac5c80a3987")        { 4 fields }                               Object
(2) ObjectId("5daa7706028acac5c80a3988")        { 6 fields }                               Object
(3) ObjectId("5daa773dc1d91667426e6db3")        { 4 fields }                               Object
(4) ObjectId("5daa773fc1d91667426e6db4")        { 6 fields }                               Object
    _id                                         ObjectId("5daa773fc1d91667426e6db4")       ObjectId
    event_dt                                    2019-10-19 02:38:55.318Z                   Date
    type                                        issue                                      String
    event                                       finish                                     String
    details                                     { 7 fields }                               Object
    start_dt                                    2019-10-19 02:38:53.900Z                   Date
```

Analytics

Once data is ingested from the Jira API to the Data Lakes, data can then be analyzed from the Data Lakes. Analytics classes should utilize the data in the Data Lakes and NOT hit the Jira API directly.



```
OPEN EDITORS
  × sprint.py src/analytics
TMP-JIRA-ANALYTICS
  > .vscode
  > mongodb
  v src
    v analytics
      > __pycache__
        __init__.py
        base.py
        sprint.py
    > common
    v data_lakes
      > __pycache__
        __init__.py
        base_type.py
        data_lake.py
        ingest_types.py
      runner.py
    __init__.py
  .docker.env
  .gitignore
  docker-compose.yml
  Dockerfile
  Makefile
  README.md
  requirements.txt
```

```
src > analytics > sprint.py > SprintAnalytics > get_unestimated_totals
You, an hour ago | 1 author (You)
"""
Functions for sprint analytics
"""
import pymongo
from analytics.base import BaseAnalytics
from data_lakes.ingest_types import Board
from data_lakes.ingest_types import Sprint
from data_lakes.ingest_types import SprintReport


You, an hour ago | 1 author (You)
class SprintAnalytics(BaseAnalytics):
    """ SprintAnalytics class """

    def get_last_sprint_metrics_report(self, project_key, count=1):
        """ Get's the last sprint metrics for the specified project """
        boards = Board().get(
            query={'location.projectKey': project_key, 'type': 'scrum'})

        # Ensure we get exactly one board for the project
        if boards.count() != 1:
            self.logger.error('Project %s has %s scrum boards.  Should be 1.',
                              project_key, boards.count())
            return []
        board_id = boards[0]['id']

        # Get sprints for board
        sprints = Sprint().collection.find(
            {'originBoardId': board_id, 'state': 'closed'}
            ).sort('endDate', pymongo.DESCENDING).limit(count)
```

Analytics utilize Data Lake classes to pull data from the database.
Example query from Data Lake