Building scalable and robust Data Science solutions

A Developer's perspective

About The Author

- Bachelor in Comp Sc
- Worked as a developer for several years
- Master in Information management systems (With DS as major)
- Currently working as Director of Machine learning, Analytics in ABInBev

- Work Ex

- Education

Before we start, let's look at reddit -

"the DS's poor excuse for a pipeline The DS pipeline: run jupyter notebook 1, copy the result and paste it into notebook 2 as a variable."

"Wtf man. I swear that whenever I ask them to push the code to git and I'll take a look. They literally push a notebook."

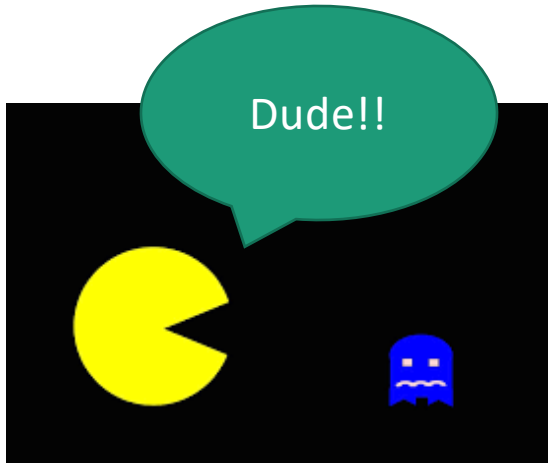
"My org is very similar but the ML and Eng teams are also separate. Also the ML people are called "data scientists" and the DS people are called "decision scientists" so it's not at all confusing.."

"Fortunately our Scientist bands are higher than Engineering bands, but we hire mostly PhDs who can all code to some degree, and code as well as software engineers sometimes."

Reddit Source

No Ankit, that happens very rarely…

"We are working on a Next Gen AI that would make Skynet look like Pacman"




```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": [
    "## Import the dependencies\n",
    "import tabulate\n",
    "import pandas as pd\n",
    "import numpy as np\n",
    "import itertools\n",
    "import re\n",
    "import datetime\n",
    "import matplotlib.pyplot as plt\n",
    "import math\n",
    "from sklearn import datasets, linear_model, metrics \n",
    "import seaborn as sns\n",
    "from sklearn.ensemble import RandomForestRegressor\n",
    "from sklearn.linear_model import LinearRegression\n",
    "import statsmodels.api as sm\n",
    "from scipy import stats\n",
    "from azure.storage.blob import *\n",
    "from io import StringIO"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {},
```

Ankit, stop being a counterproductive bunch...

Problem -

"We want to use an OS python library but customize around 5 lines of the code"

-- New joiners in the team

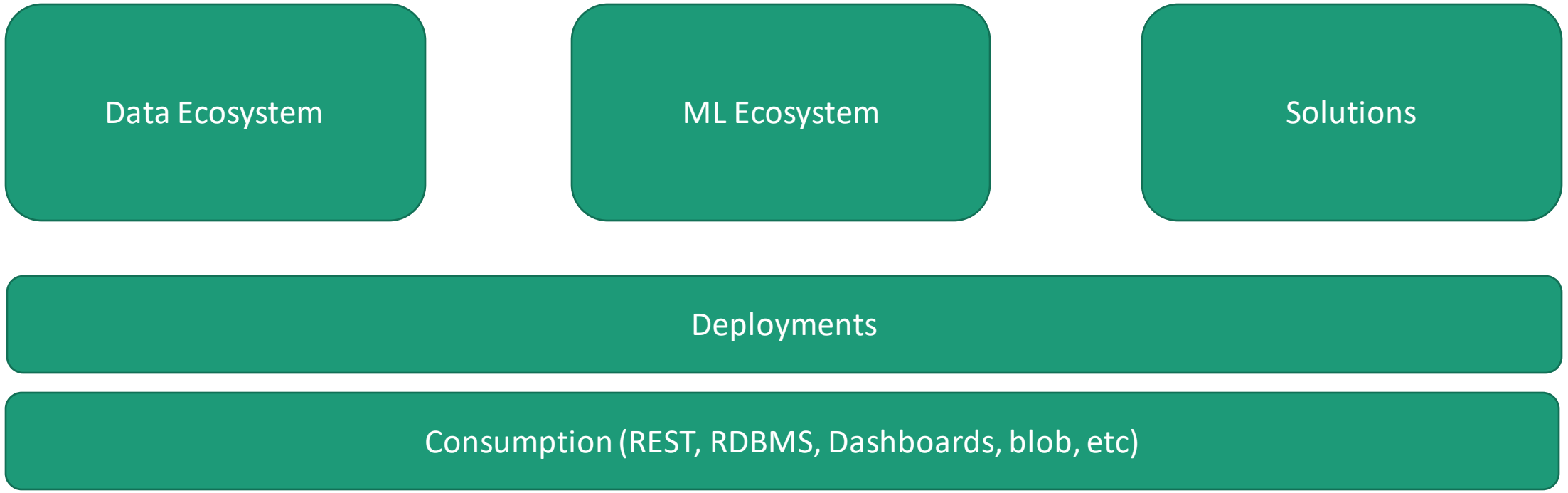
Solution -

"Take 500 lines of code out from the upstream package relevant to us, copy and paste those into a notebook, change the five lines that you need"

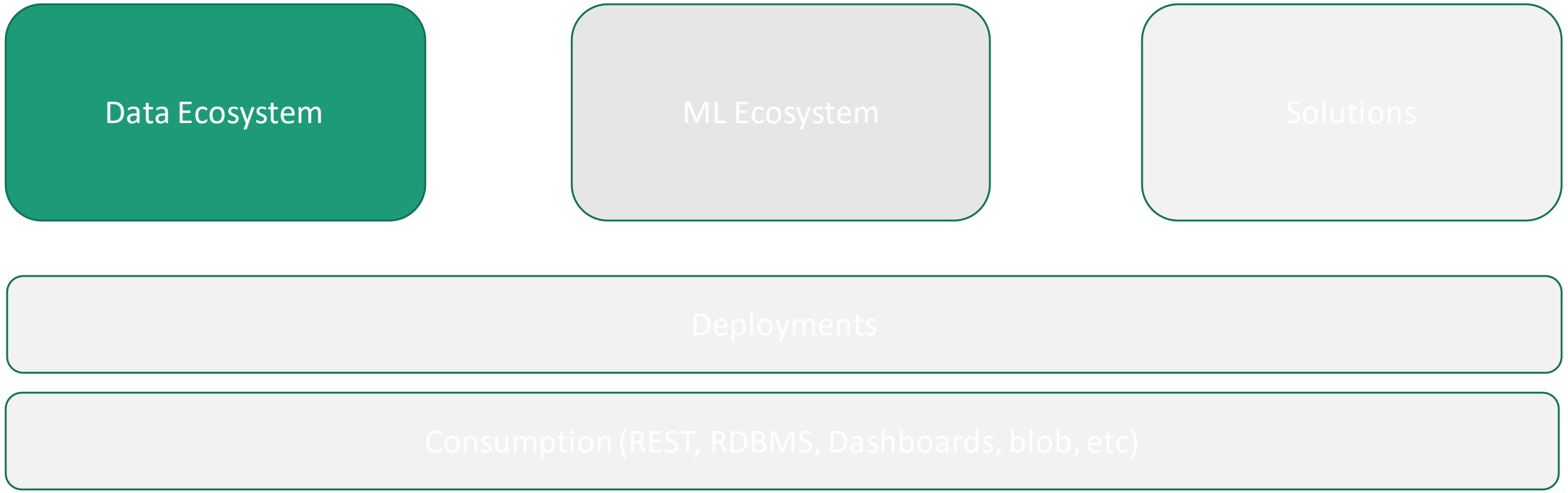
You ask why?

Cause "results" and "ease" of getting there are important

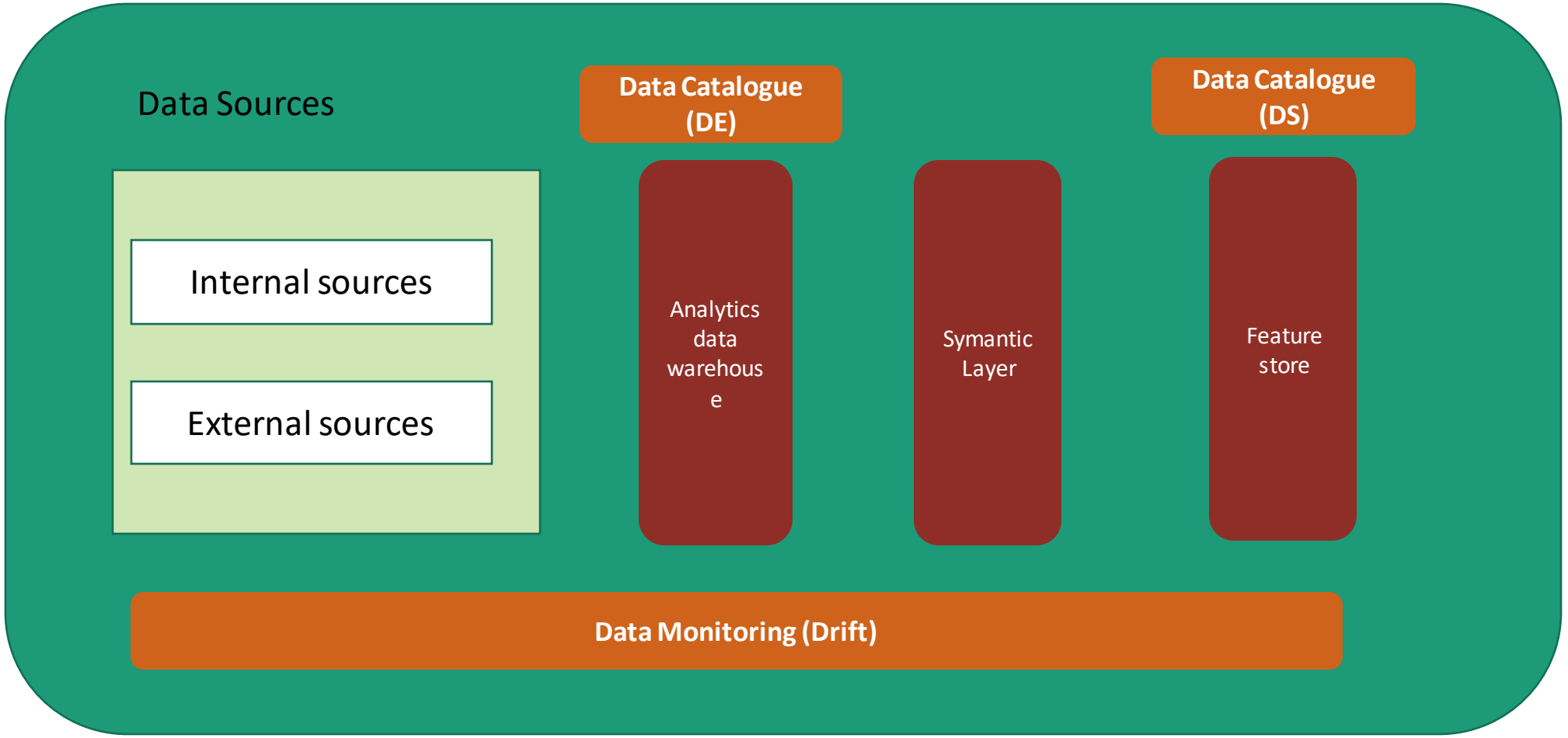
Being Production ready



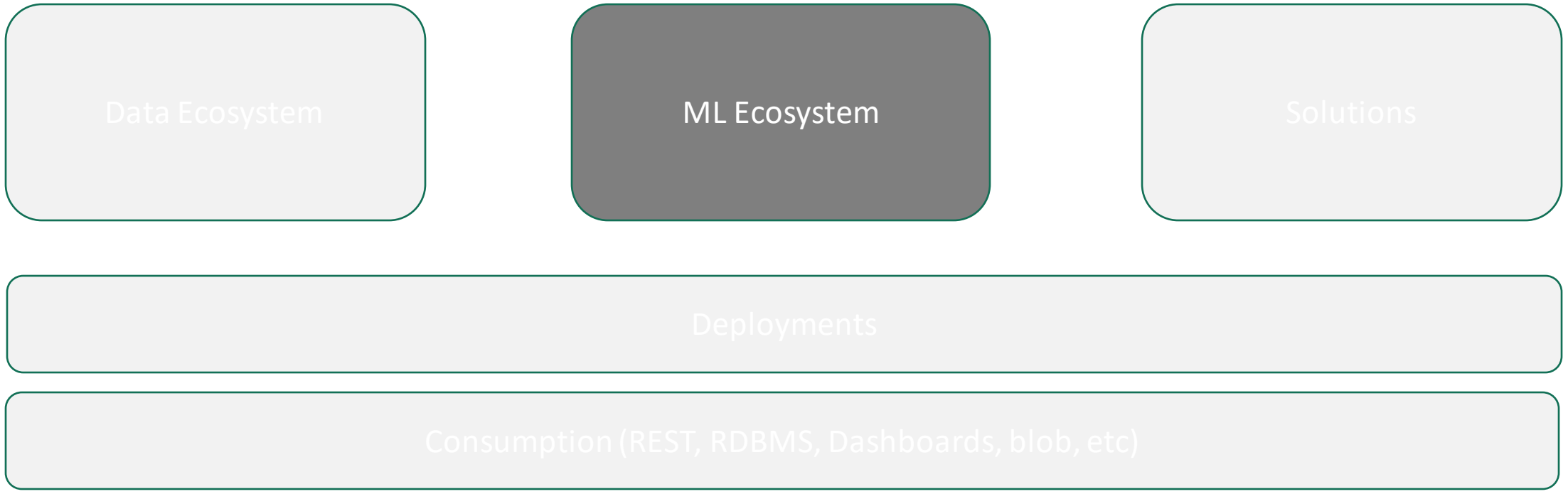
Being Production ready



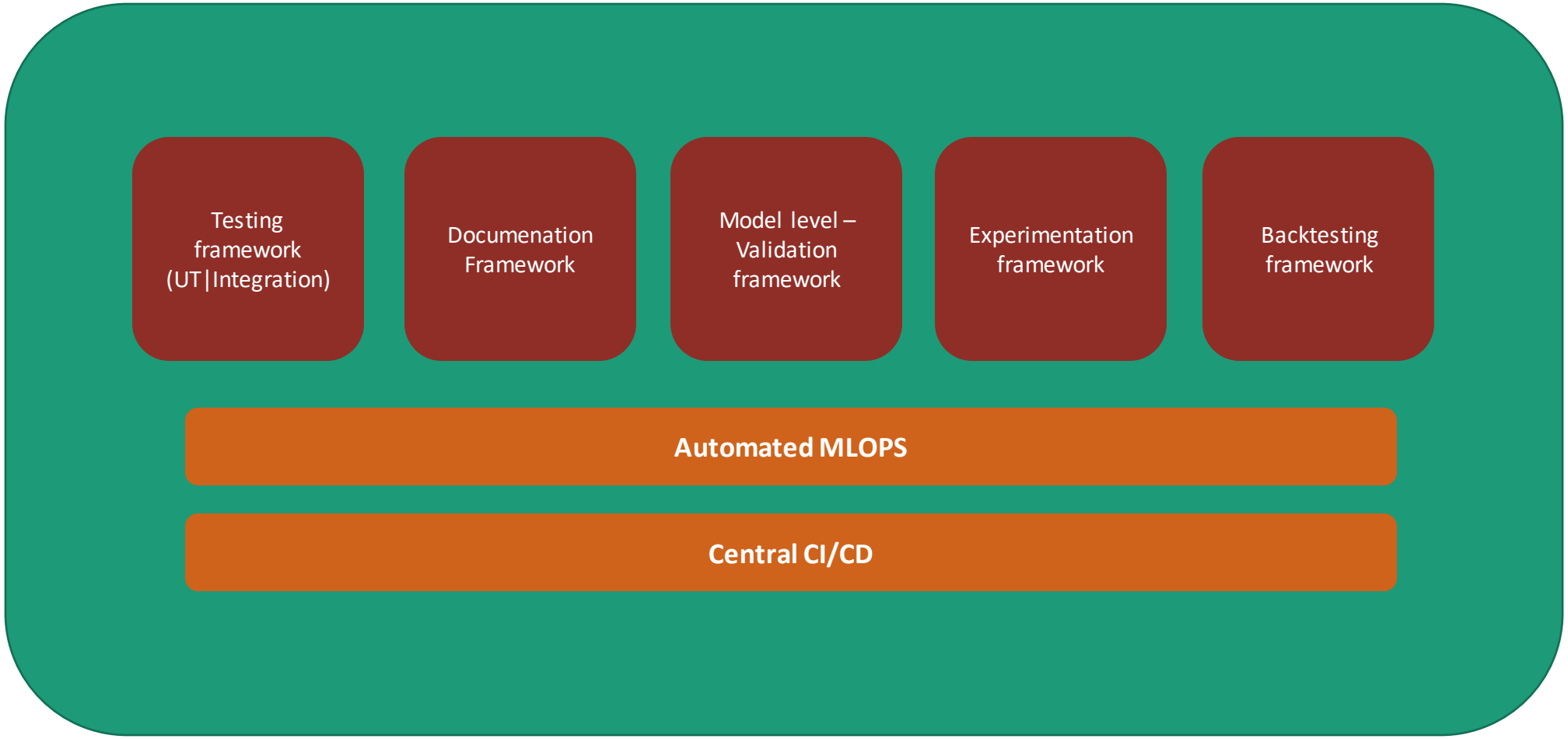
Data Ecosystem



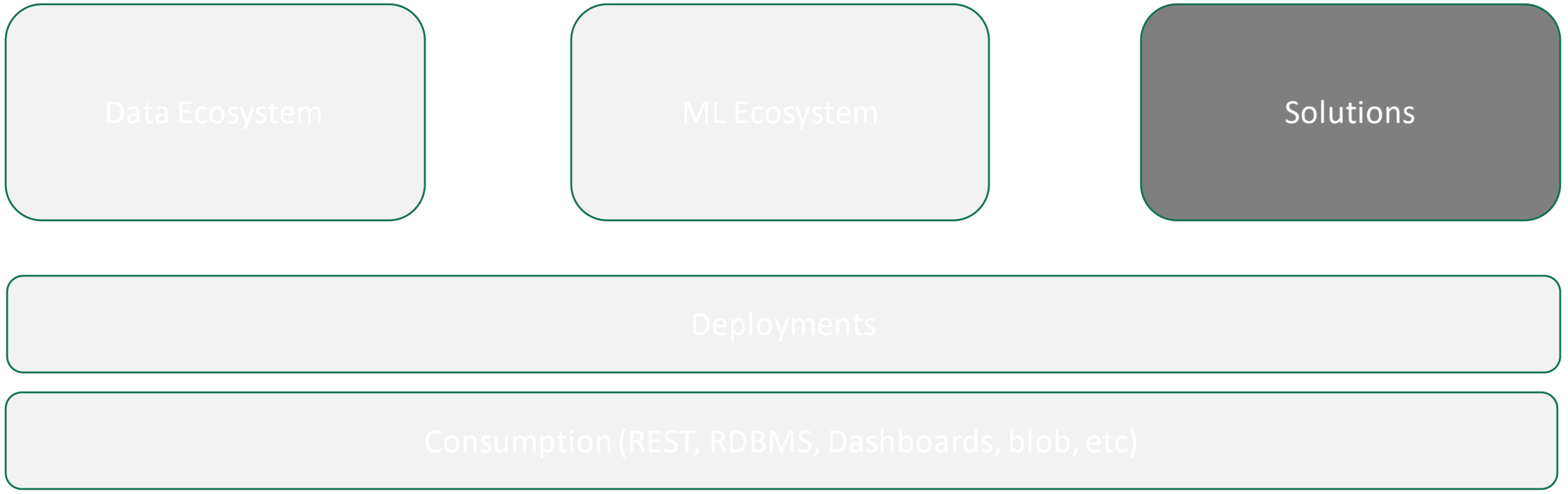
Being Production ready



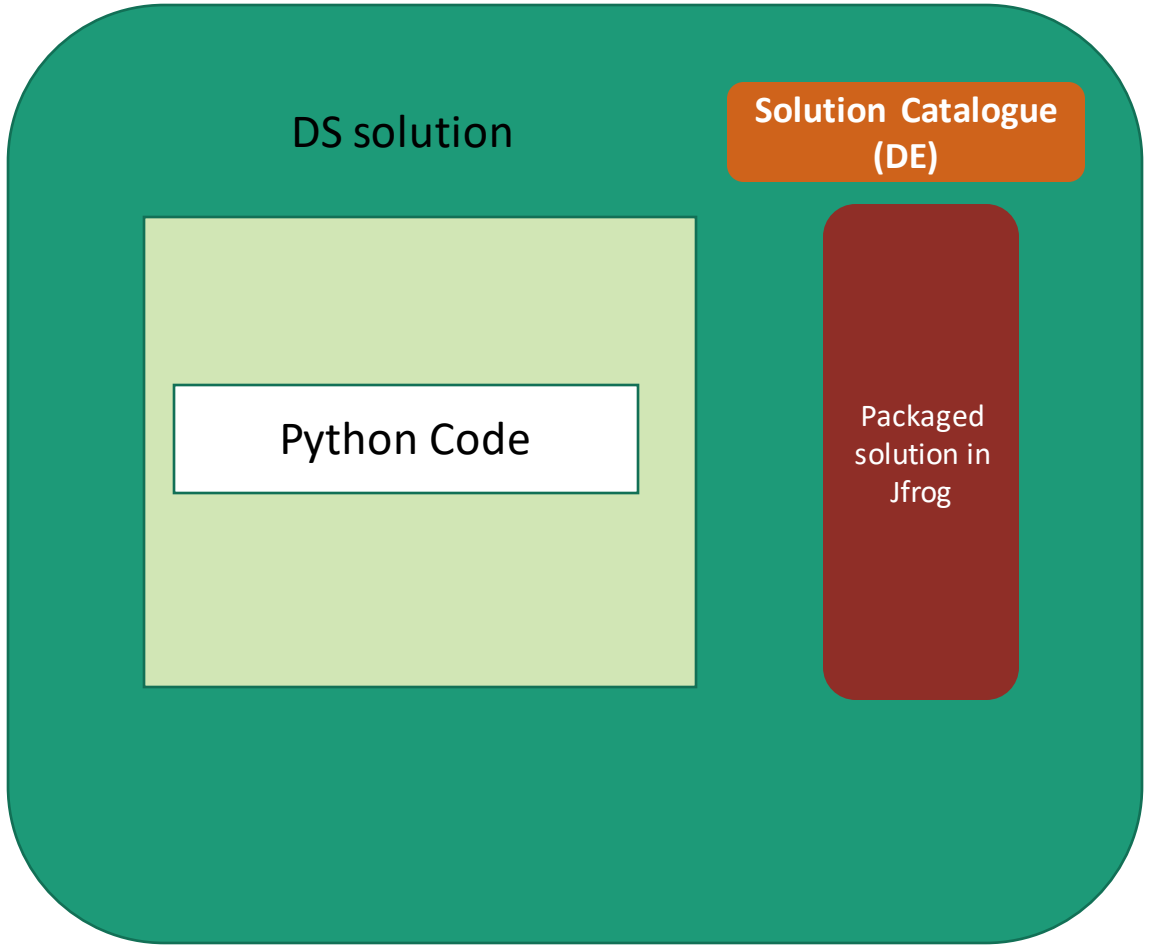
ML Ecosystem



Being Production ready



Solution


DS solution
Solution Catalogue (DE)
Python Code
Packaged solution in Jfrog

Building packages? Isn't that simple?

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4,
                           n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)
print(clf.predict([[0, 0, 0, 0]]))
```

Let's break it down..

User View -
- Clean APIs for the user
- Excellent examples and documentations
- Reliable output
- Readability
- Stable APIs (Backwards compatibility)

Design View -
- Architecture that is easy to develop and future compatible
- Coding standards – pep8/484
- Development that treats backward compatibility highly
- Extensive Unit tests
- Tests that go beyond Unit tests - How do we reliably test a process that has randomness ?

Process View -
- Release cadence
- Licenses
- Bug reporting/contributing
- Announcements

Infra View -
- Platforms supported
- Private/public package
- Masking code
- Sharing mechanism

Building the package

```
setup(
    ext_modules=generate_extensions(),
    name="baby_skynet",
    version="1.0",
    author="Arnold",
    author_email="arnold@has.to.come",
    description="A package for reaching Transcendence",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/ankitagarwal/pycon_2020",
    license="GPL3",
    packages=[],
    classifiers=[
        "Programming Language :: Python :: 3.7",
        "Intended Audience :: Developers",
        "License :: Other/Proprietary License",
        "Operating System :: Microsoft :: Windows",
        "Operating System :: LINUX",
        "Operating System :: Apple :: MacOS",
        "Topic :: Scientific/Engineering",
        "Topic :: Scientific/Engineering :: Mathematics",
        "Topic :: Scientific/Engineering :: Artificial Intelligence",
        "Topic :: Software Development :: Libraries :: Python Modules",
    ],
    platforms=["Windows", "Linux", "MacOS"],
    test_suite="pytest",
    python_requires='>=3.7',
    data_files=[
        ('baby-skynet', ['README.md']),
        ('baby-skynet', ['requirements.txt']),
    ],
)
```

Handling Auhtentication against Private Artifactory Server

Update *.pypirc* file in home Directory with the following

```
[distutils]
index-servers = local
[local]
repository: <URL>

username: <USERNAME>
password: <PASSWORD>
```

Deploying a version of the package

As a wheel -

python setup.py bdist_wheel upload -r local

With source -

python setup.py sdist upload -r local

Installing the package

Option 1

pip install baby_skynet --extra-index-url=https://user:token@<server>

Option 2

Create a .netrc file with following content

Machine <server>

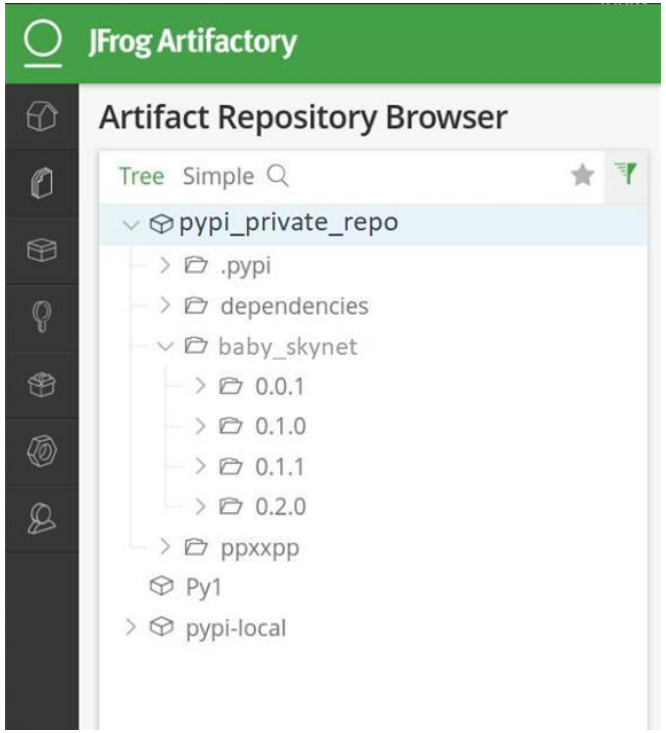
Login <username>

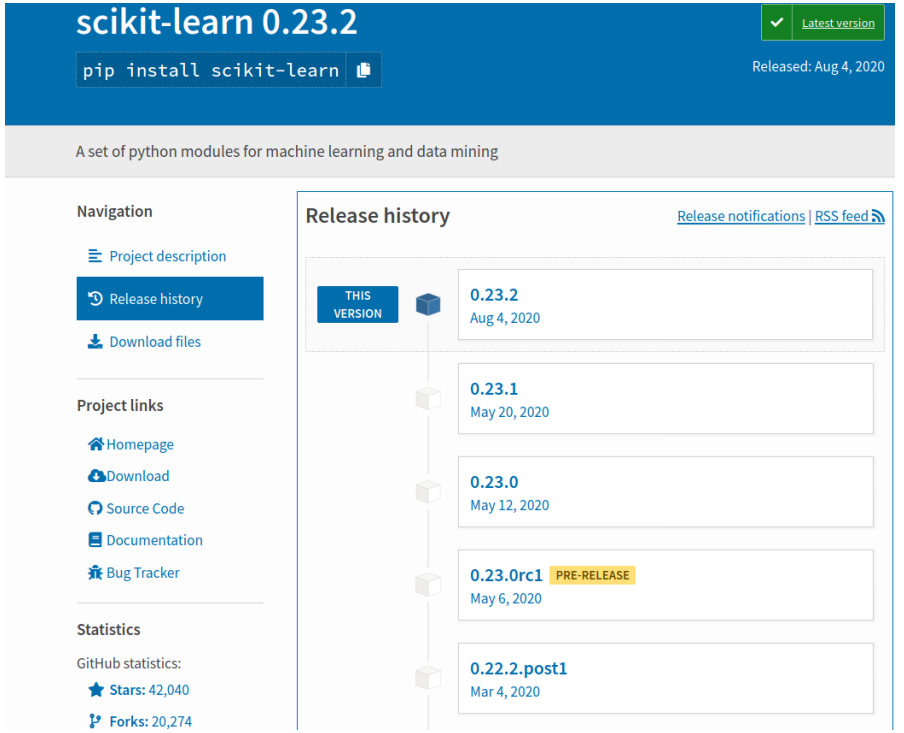
Password <token>

pip install baby_skynet --extra-index-url=<server>

Finally we are there..

```
Python 3.7.6 (default, Jan  8 2020, 19:59:22)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from baby_skynet import Skynet
>>> Skynet()
Skynet is still a baby and knows nothing, please build it more.
<baby_skynet.Skynet.Skynet object at 0x7f7fb50ec810>
>>>
```





Deployments

Use-case-country

ML pipeline (AML|ADB)

Other supporting files

Docker pre-built with 'solution package'

YAML Copy

```yaml
pipeline:
    name: SamplePipelineFromYaml
    parameters:
        PipelineParam1:
            type: int
            default: 3
    data_references:
        adls_test_data:
            datastore: adftestadla
            path_on_datastore: "testdata"
        blob_test_data:
            datastore: workspaceblobstore
            path_on_datastore: "dbtest"
    default_compute: mydatabricks
    steps:
        Step1:
            runconfig: "D:\\Yaml\\default_runconfig.yml"
            parameters:
                NUM_ITERATIONS_2:
                    source: PipelineParam1
                NUM_ITERATIONS_1: 7
            type: "DatabricksStep"
            name: "MyDatabrickStep"
            run_name: "DatabricksRun"
            python_script_name: "train-db-local.py"
            source_directory: "D:\\scripts\\Databricks"
            num_workers: 1
            allow_reuse: true
            inputs:
                blob_test_data:
                    source: blob_test_data
            outputs:
                OutputData:
                    destination: Output4
                    datastore: workspaceblobstore
                    bind_mode: mount
```

MLOPS

Link to the materials used in this presentation

Github