FUTUREWEI Technologies

Wenju: Accelerating Time to Value for Enterprise AI

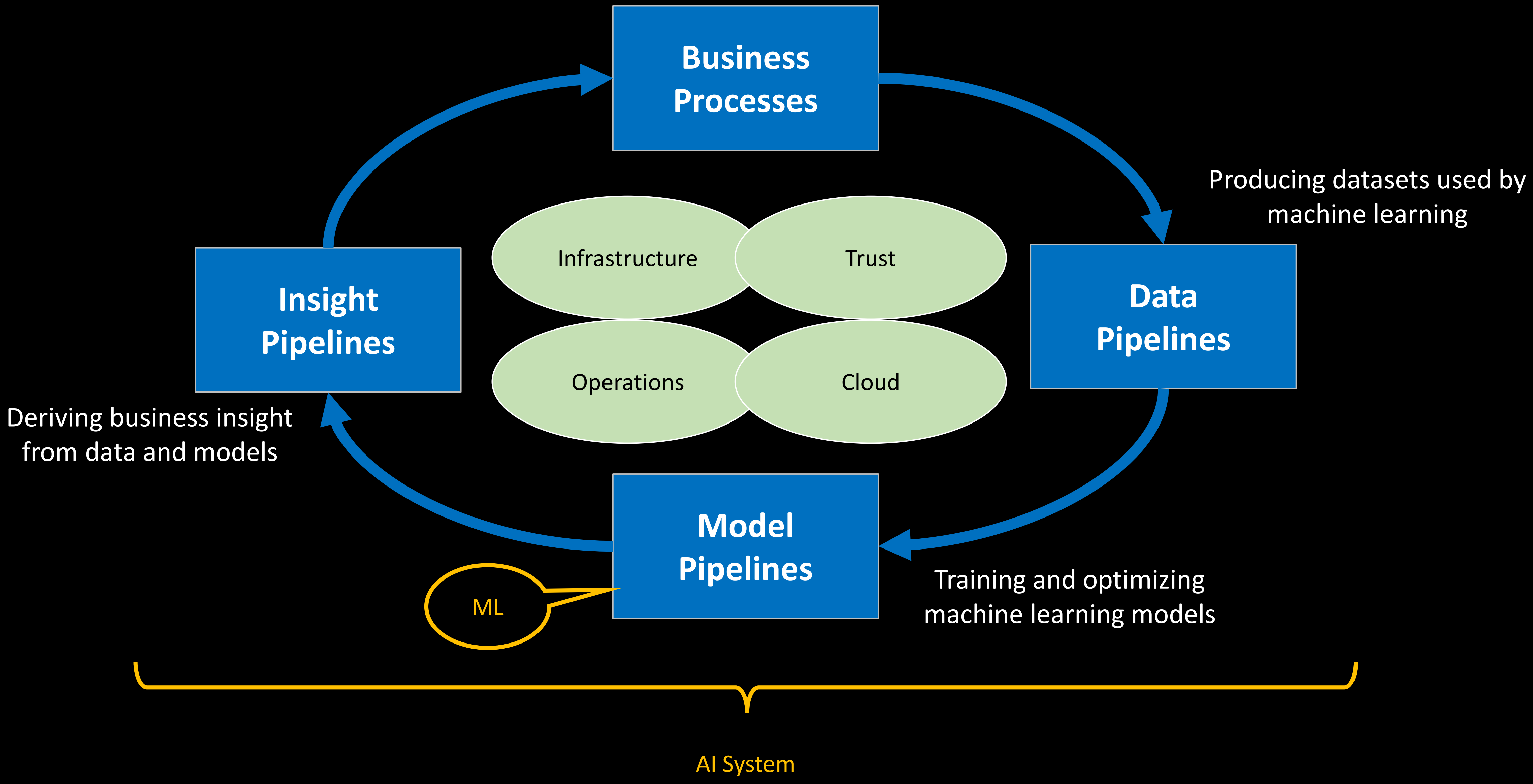
Contact: TheWenjuProj@gmail.com

The Reality of Enterprise AI

- Only 8% of enterprises adopted AI by end of 2019 (Cognilytica)
- Some companies have seen less than 10% of their AI pilots reach full-scale production (IIA)
- Just 15% of projects for AI and IoT will be successful by 2021 (Gartner)

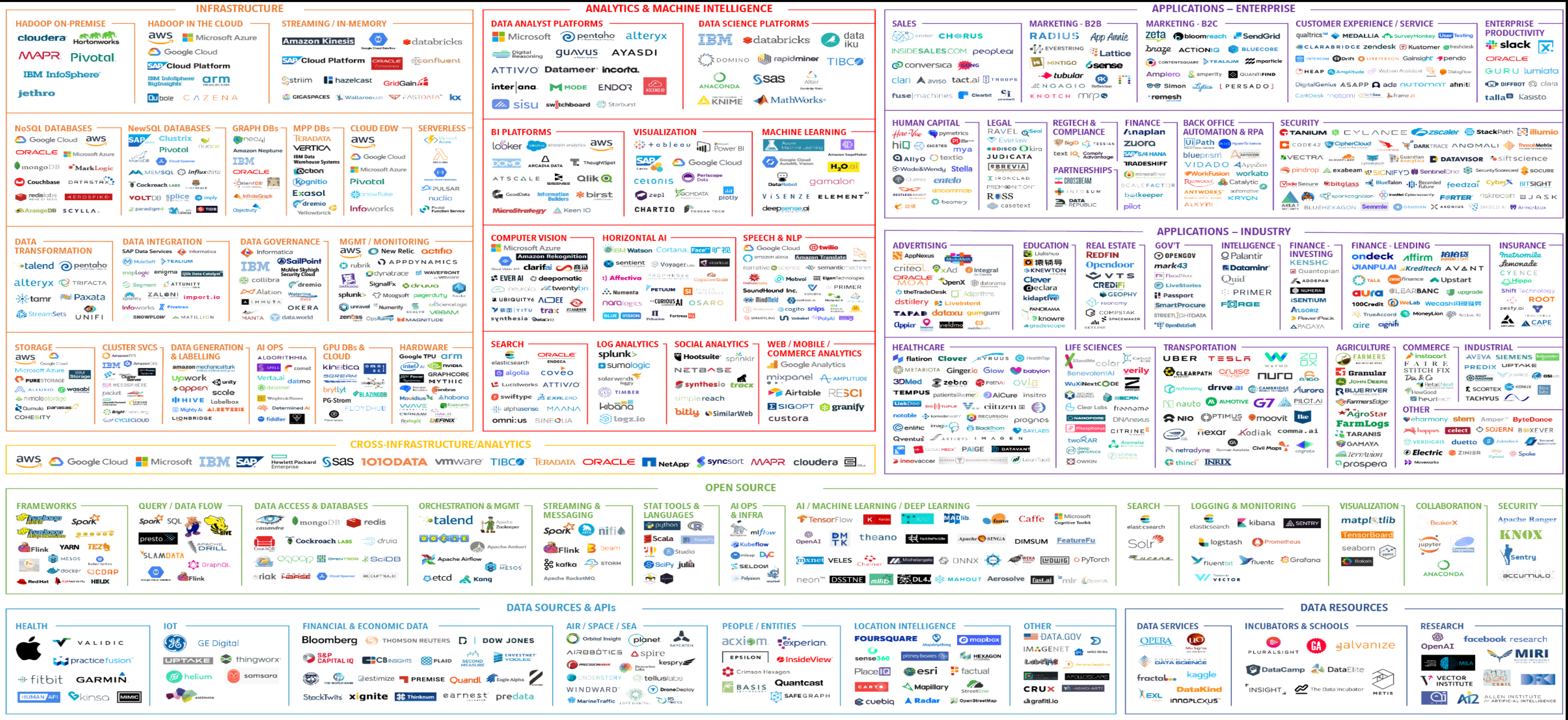
Surprise! The problem is not with ML.

AI Is a Lot More Than ML


Business Processes
Producing datasets used by machine learning
Data Pipelines
Training and optimizing machine learning models
Model Pipelines
ML
Insight Pipelines
Deriving business insight from data and models
Infrastructure
Trust
Operations
Cloud
AI System

The Proliferation of AI Tooling

- The AI tooling landscape is crowded and confusing
- Current tools require deep expertise and skills
- Integration of siloed tools incurs large overhead and technical debt



Source: https://mattturck.com/data2019

Introducing Wenju

Wenju is an open-source platform for enterprise AI solutioning

- One-stop shop for data integration, model building, and insight generation
- Collaborative environment for data engineers, data scientists, ML engineers, software engineers, and IT engineers

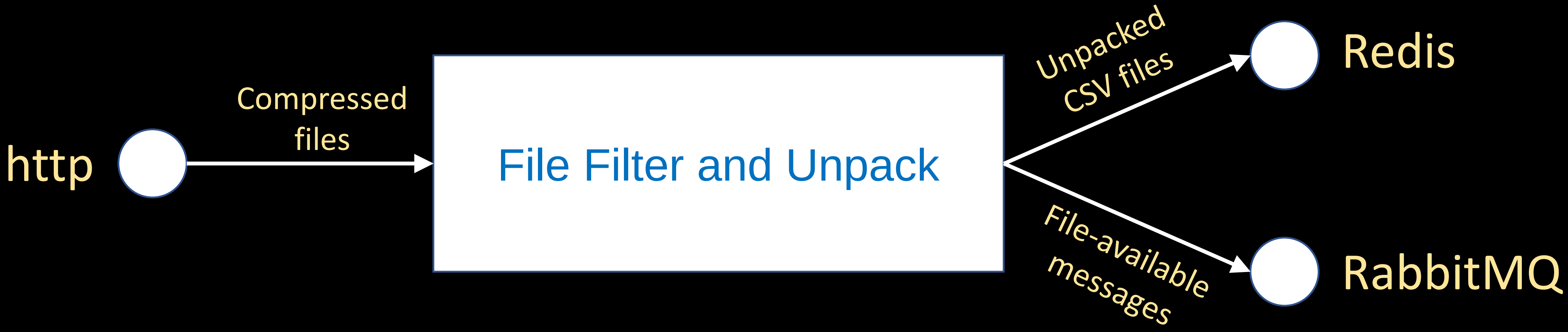
Wenju provides a low-code development platform for AI applications

- Visual programming
- Dataflow programming
- Model-driven development

The Main Concepts in Wenju

- At the center of Wenju is the notion of xProcessors
  - Building blocks that simplify the development of AI applications
  - High-level abstractions for a wide-variety of data and ML frameworks
  - Essential functions required by end-to-end AI solutions
  - Configurable and customizable
- Application developers use xProcessors to build xGraphs
  - An xGraph represents an executable pipeline of an application
- xProcessors in an xGraph are connected by ports
- Data that flows between xProcessors through ports is defined by xData

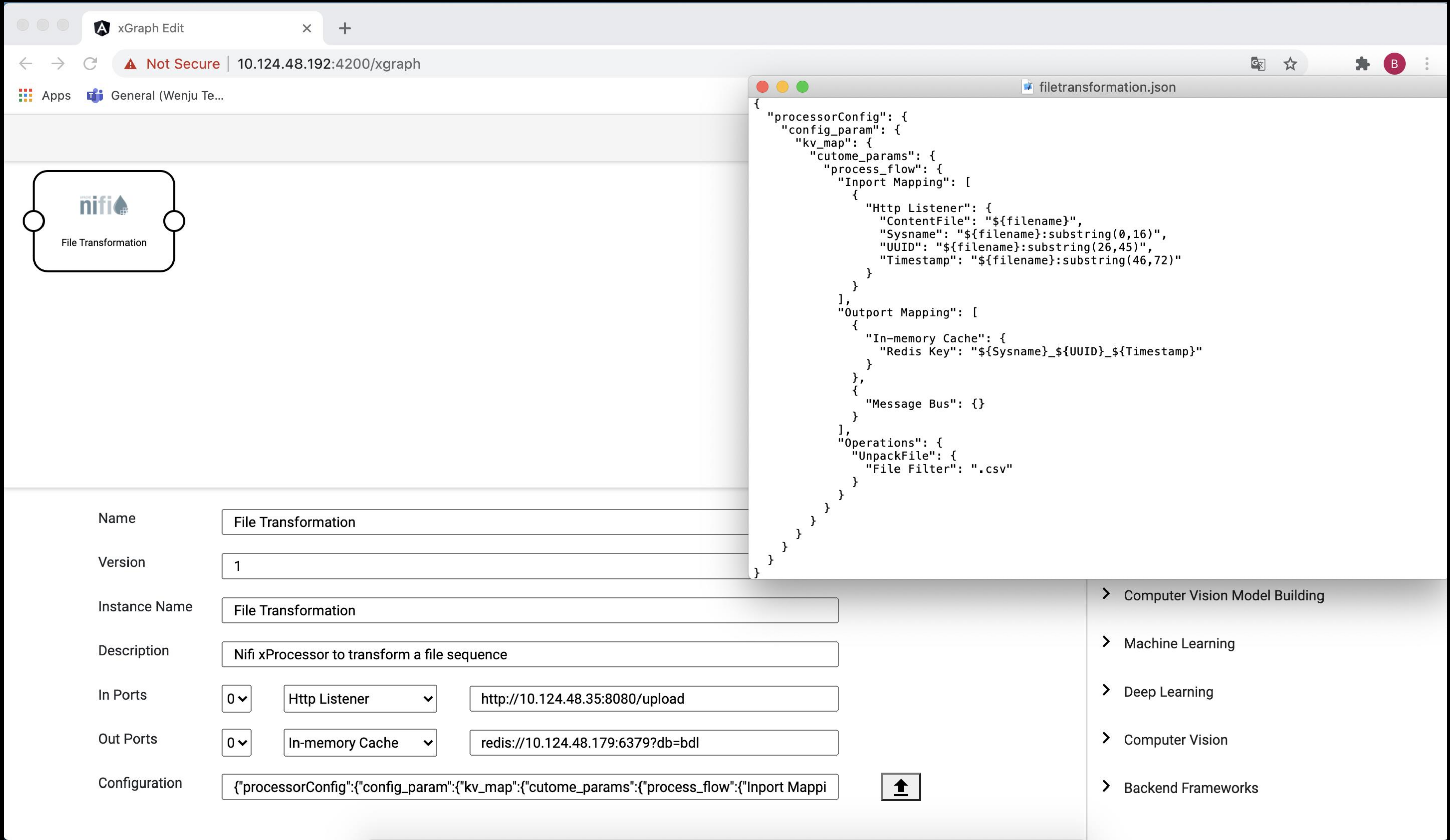
An Illustrating Example



Consider code that performs the following:

- Ingest a stream of compressed files through http
- Select .csv files and unpack them
- Deposit the unpacked files in a Redis database
- Post a message via RabbitMQ whenever an unpacked file is available

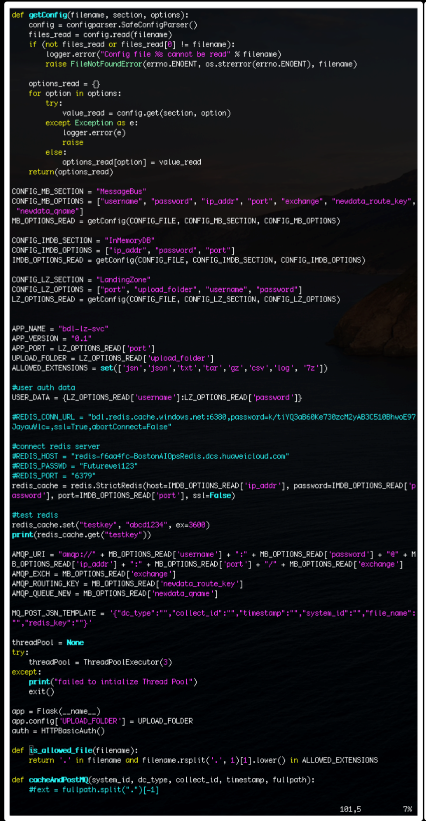
The Wenju Way



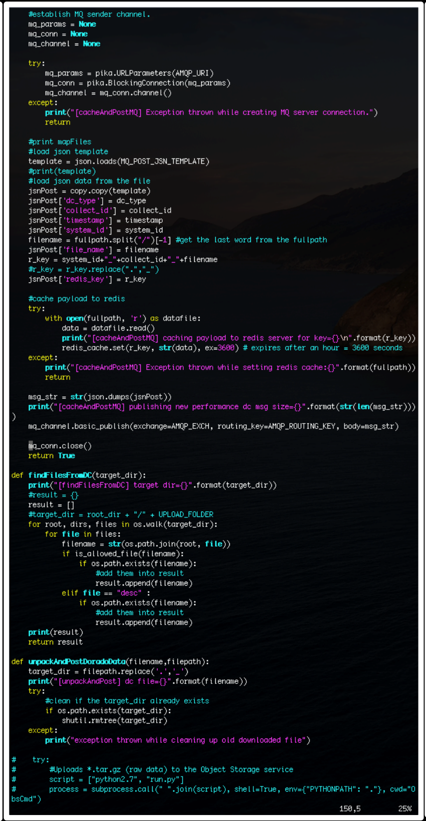
```
{
  "processorConfig": {
    "config_param": {
      "kv_map": {
        "cutome_params": {
          "process_flow": {
            "Inport Mapping": [
              {
                "Http Listener": {
                  "ContentFile": "${filename}",
                  "Sysname": "${filename}:substring(0,16)",
                  "UUID": "${filename}:substring(26,45)",
                  "Timestamp": "${filename}:substring(46,72)"
                }
              }
            ],
            "Outport Mapping": [
              {
                "In-memory Cache": {
                  "Redis Key": "${Sysname}_${UUID}_${Timestamp}"
                }
              },
              {
                "Message Bus": {}
              }
            ],
            "Operations": {
              "UnpackFile": {
                "File Filter": ".csv"
              }
            }
          }
        }
      }
    }
  }
}
```

Name	File Transformation	
Version	1	
Instance Name	File Transformation	
Description	Nifi xProcessor to transform a file sequence	
In Ports	0 Http Listener	http://10.124.48.35:8080/upload
Out Ports	0 In-memory Cache	redis://10.124.48.179:6379?db=bdl
Configuration	{"processorConfig":{"config_param":{"kv_map":{"cutome_params":{"process_flow":{"Inport Mappi	

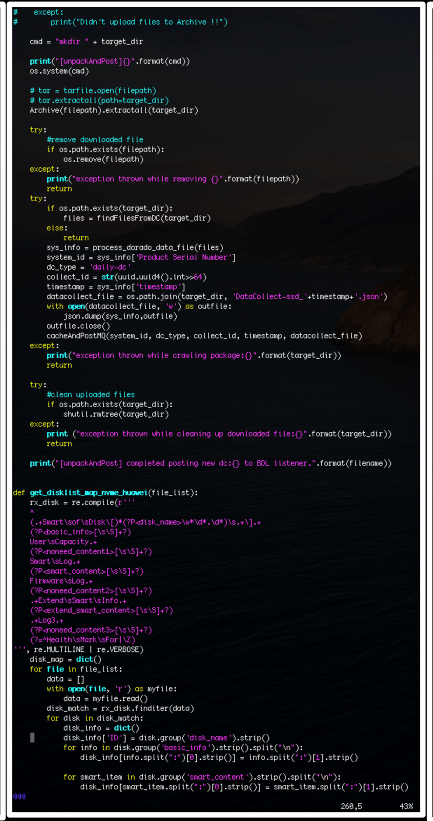
In Comparison: the Traditional Way



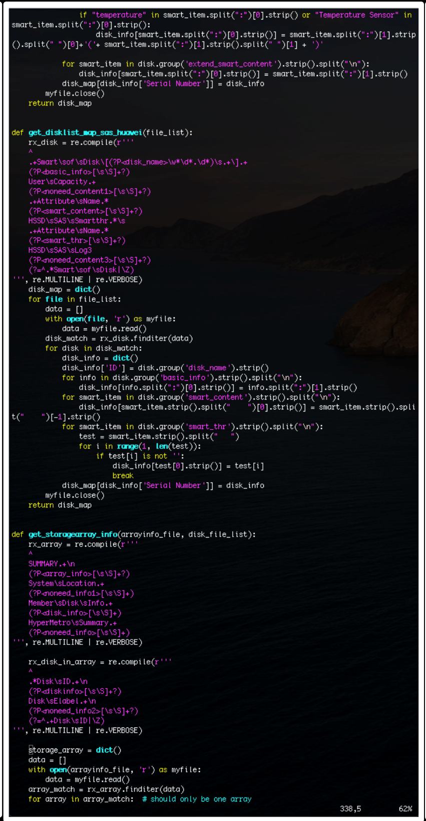
```python
def getConfig(filename, section, options):
    config = configparser.SafeConfigParser()
    files_read = config.read(filename)
    if (not files_read or files_read[0] != filename):
        logger.error("Config file %s cannot be read" % filename)
        raise FileNotFoundError(errno.ENOENT, os.strerror(errno.ENOENT), filename)

    options_read = {}
    for option in options:
        try:
            value_read = config.get(section, option)
        except Exception as e:
            logger.error(e)
            raise
        else:
            options_read[option] = value_read
    return(options_read)

CONFIG_MB_SECTION = "MessageBus"
CONFIG_MB_OPTIONS = ["username", "password", "ip_addr", "port", "exchange", "newdata_route_key",
 "newdata_qname"]
MB_OPTIONS_READ = getConfig(CONFIG_FILE, CONFIG_MB_SECTION, CONFIG_MB_OPTIONS)

CONFIG_IMDB_SECTION = "InMemoryDB"
CONFIG_IMDB_OPTIONS = ["ip_addr", "password", "port"]
IMDB_OPTIONS_READ = getConfig(CONFIG_FILE, CONFIG_IMDB_SECTION, CONFIG_IMDB_OPTIONS)

CONFIG_LZ_SECTION = "LandingZone"
CONFIG_LZ_OPTIONS = ["port", "upload_folder", "username", "password"]
LZ_OPTIONS_READ = getConfig(CONFIG_FILE, CONFIG_LZ_SECTION, CONFIG_LZ_OPTIONS)


APP_NAME = "bdl-lz-svc"
APP_VERSION = "0.1"
APP_PORT = LZ_OPTIONS_READ['port']
UPLOAD_FOLDER = LZ_OPTIONS_READ['upload_folder']
ALLOWED_EXTENSIONS = set(['jsn','json','txt','tar','gz','csv','log', '7z'])

#user auth data
USER_DATA = {LZ_OPTIONS_READ['username']:LZ_OPTIONS_READ['password']}

#REDIS_CONN_URL = "bdl.redis.cache.windows.net:6380,password=k/tiYQ3aB60Ke730zcM2yAB3C510BhwoE97
JayauWlc=,ssl=True,abortConnect=False"

#connect redis server
#REDIS_HOST = "redis-f6aa4fc-BostonAIOpsRedis.dcs.huaweicloud.com"
#REDIS_PASSWD = "Futurewei123"
#REDIS_PORT = "6379"
redis_cache = redis.StrictRedis(host=IMDB_OPTIONS_READ['ip_addr'], password=IMDB_OPTIONS_READ['p
assword'], port=IMDB_OPTIONS_READ['port'], ssl=False)

#test redis
redis_cache.set("testkey", "abcd1234", ex=3600)
print(redis_cache.get("testkey"))

AMQP_URI = "amqp://" + MB_OPTIONS_READ['username'] + ":" + MB_OPTIONS_READ['password'] + "@" + M
B_OPTIONS_READ['ip_addr'] + ":" + MB_OPTIONS_READ['port'] + "/" + MB_OPTIONS_READ['exchange']
AMQP_EXCH = MB_OPTIONS_READ['exchange']
AMQP_ROUTING_KEY = MB_OPTIONS_READ['newdata_route_key']
AMQP_QUEUE_NEW = MB_OPTIONS_READ['newdata_qname']

MQ_POST_JSN_TEMPLATE = '{"dc_type":"","collect_id":"","timestamp":"","system_id":"","file_name":
"","redis_key":""}'

threadPool = None
try:
    threadPool = ThreadPoolExecutor(3)
except:
    print("failed to intialize Thread Pool")
    exit()

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
auth = HTTPBasicAuth()

def is_allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def cacheAndPostMQ(system_id, dc_type, collect_id, timestamp, fullpath):
    #fext = fullpath.split(".")[-1]
                                                            101,5          7%
```



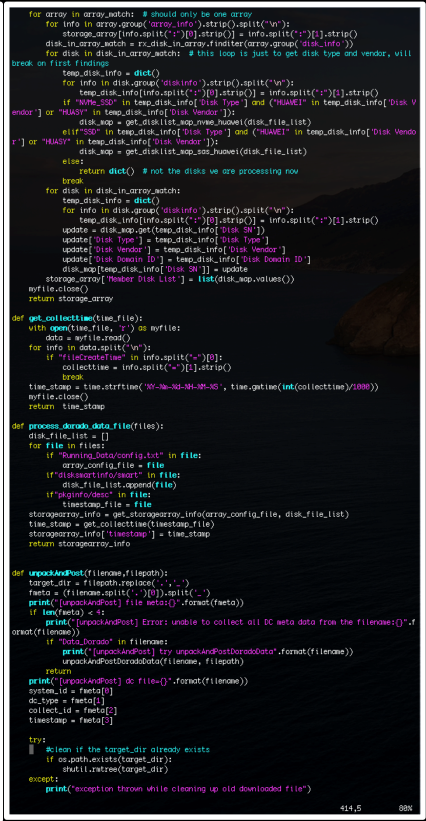
```python
    #establish MQ sender channel.
    mq_params = None
    mq_conn = None
    mq_channel = None

    try:
        mq_params = pika.URLParameters(AMQP_URI)
        mq_conn = pika.BlockingConnection(mq_params)
        mq_channel = mq_conn.channel()
    except:
        print("[cacheAndPostMQ] Exception thrown while creating MQ server connection.")
        return

    #print mapFiles
    #load json template
    template = json.loads(MQ_POST_JSN_TEMPLATE)
    #print(template)
    #load json data from the file
    jsnPost = copy.copy(template)
    jsnPost['dc_type'] = dc_type
    jsnPost['collect_id'] = collect_id
    jsnPost['timestamp'] = timestamp
    jsnPost['system_id'] = system_id
    filename = fullpath.split("/")[-1] #get the last word from the fullpath
    jsnPost['file_name'] = filename
    r_key = system_id+"_"+collect_id+"_"+filename
    #r_key = r_key.replace(".","_")
    jsnPost['redis_key'] = r_key

    #cache payload to redis
    try:
        with open(fullpath, 'r') as datafile:
            data = datafile.read()
            print("[cacheAndPostMQ] caching payload to redis server for key={}\n".format(r_key))
            redis_cache.set(r_key, str(data), ex=3600) # expires after an hour = 3600 seconds
    except:
        print("[cacheAndPostMQ] Exception thrown while setting redis cache:{}".format(fullpath))
        return

    msg_str = str(json.dumps(jsnPost))
    print("[cacheAndPostMQ] publishing new performance dc msg size={}".format(str(len(msg_str)))
)
    mq_channel.basic_publish(exchange=AMQP_EXCH, routing_key=AMQP_ROUTING_KEY, body=msg_str)

    mq_conn.close()
    return True

def findFilesFromDC(target_dir):
    print("[findFilesFromDC] target dir={}".format(target_dir))
    #result = {}
    result = []
    #target_dir = root_dir + "/" + UPLOAD_FOLDER
    for root, dirs, files in os.walk(target_dir):
        for file in files:
            filename = str(os.path.join(root, file))
            if is_allowed_file(filename):
                if os.path.exists(filename):
                    #add them into result
                    result.append(filename)
            elif file == "desc" :
                if os.path.exists(filename):
                    #add them into result
                    result.append(filename)
    print(result)
    return result

def unpackAndPostDoradoData(filename,filepath):
    target_dir = filepath.replace('.','_')
    print("[unpackAndPost] dc file={}".format(filename))
    try:
        #clean if the target_dir already exists
        if os.path.exists(target_dir):
            shutil.rmtree(target_dir)
    except:
        print("exception thrown while cleaning up old downloaded file")

#    try:
#        #Uploads *.tar.gz (raw data) to the Object Storage service
#        script = ["python2.7", "run.py"]
#        process = subprocess.call(" ".join(script), shell=True, env={"PYTHONPATH": "."}, cwd="O
bsCmd")
                                                            150,5         25%
```



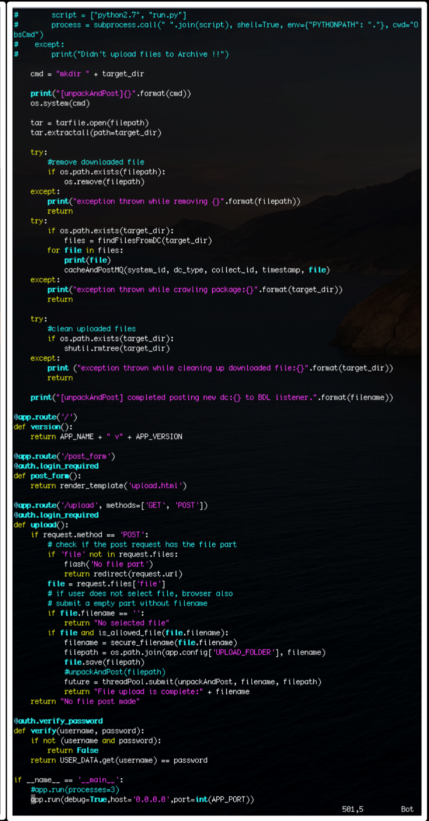
```python
#    except:
#        print("Didn't upload files to Archive !!")

    cmd = "mkdir " + target_dir

    print("[unpackAndPost]{}".format(cmd))
    os.system(cmd)

    # tar = tarfile.open(filepath)
    # tar.extractall(path=target_dir)
    Archive(filepath).extractall(target_dir)

    try:
        #remove downloaded file
        if os.path.exists(filepath):
            os.remove(filepath)
    except:
        print("exception thrown while removing {}".format(filepath))
        return
    try:
        if os.path.exists(target_dir):
            files = findFilesFromDC(target_dir)
        else:
            return
        sys_info = process_dorado_data_file(files)
        system_id = sys_info['Product Serial Number']
        dc_type = 'daily-dc'
        collect_id = str(uuid.uuid4().int>>64)
        timestamp = sys_info['timestamp']
        datacollect_file = os.path.join(target_dir, 'DataCollect-ssd_'+timestamp+'.json')
        with open(datacollect_file, 'w') as outfile:
            json.dump(sys_info,outfile)
        outfile.close()
        cacheAndPostMQ(system_id, dc_type, collect_id, timestamp, datacollect_file)
    except:
        print("exception thrown while crawling package:{}".format(target_dir))
        return

    try:
        #clean uploaded files
        if os.path.exists(target_dir):
            shutil.rmtree(target_dir)
    except:
        print ("exception thrown while cleaning up downloaded file:{}".format(target_dir))
        return

    print("[unpackAndPost] completed posting new dc:{} to BDL listener.".format(filename))


def get_disklist_map_nvme_huawei(file_list):
    rx_disk = re.compile(r'''
    ^
    (.+Smart\sof\sDisk\[)*(?P<disk_name>\w*\d*.\d*)\s.+\].+
    (?P<basic_info>[\s\S]+?)
    User\sCapacity.+
    (?P<noneed_content1>[\s\S]+?)
    Smart\sLog.+
    (?P<smart_content>[\s\S]+?)
    Firmware\sLog.+
    (?P<noneed_content2>[\s\S]+?)
    .+Extend\sSmart\sInfo.+
    (?P<extend_smart_content>[\s\S]+?)
    .+Log3.+
    (?P<noneed_content3>[\s\S]+?)
    (?=^Health\sMark\sFor|\Z)
''', re.MULTILINE | re.VERBOSE)
    disk_map = dict()
    for file in file_list:
        data = []
        with open(file, 'r') as myfile:
            data = myfile.read()
        disk_match = rx_disk.finditer(data)
        for disk in disk_match:
            disk_info = dict()
            disk_info['ID'] = disk.group('disk_name').strip()
            for info in disk.group('basic_info').strip().split("\n"):
                disk_info[info.split(":")[0].strip()] = info.split(":")[1].strip()

            for smart_item in disk.group('smart_content').strip().split("\n"):
                disk_info[smart_item.split(":")[0].strip()] = smart_item.split(":")[1].strip()
@@@
                                                            260,5         43%
```



```python
                if "temperature" in smart_item.split(":")[0].strip() or "Temperature Sensor" in
smart_item.split(":")[0].strip():
                    disk_info[smart_item.split(":")[0].strip()] = smart_item.split(":")[1].strip
().split(" ")[0]+'('+ smart_item.split(":")[1].strip().split(" ")[1] + ')'

            for smart_item in disk.group('extend_smart_content').strip().split("\n"):
                disk_info[smart_item.split(":")[0].strip()] = smart_item.split(":")[1].strip()
            disk_map[disk_info['Serial Number']] = disk_info
        myfile.close()
    return disk_map


def get_disklist_map_sas_huawei(file_list):
    rx_disk = re.compile(r'''
    ^
    .+Smart\sof\sDisk\[(?P<disk_name>\w*\d*.\d*)\s.+\].+
    (?P<basic_info>[\s\S]+?)
    User\sCapacity.+
    (?P<noneed_content1>[\s\S]+?)
    .+Attribute\sName.*
    (?P<smart_content>[\s\S]+?)
    HSSD\sSAS\sSmartthr.*\s
    .+Attribute\sName.*
    (?P<smart_thr>[\s\S]+?)
    HSSD\sSAS\sLog3
    (?P<noneed_content3>[\s\S]+?)
    (?=^.*Smart\sof\sDisk|\Z)
''', re.MULTILINE | re.VERBOSE)
    disk_map = dict()
    for file in file_list:
        data = []
        with open(file, 'r') as myfile:
            data = myfile.read()
        disk_match = rx_disk.finditer(data)
        for disk in disk_match:
            disk_info = dict()
            disk_info['ID'] = disk.group('disk_name').strip()
            for info in disk.group('basic_info').strip().split("\n"):
                disk_info[info.split(":")[0].strip()] = info.split(":")[1].strip()
            for smart_item in disk.group('smart_content').strip().split("\n"):
                disk_info[smart_item.strip().split("    ")[0].strip()] = smart_item.strip().spli
t("    ")[-1].strip()
            for smart_item in disk.group('smart_thr').strip().split("\n"):
                test = smart_item.strip().split("   ")
                for i in range(1, len(test)):
                    if test[i] is not '':
                        disk_info[test[0].strip()] = test[i]
                        break
            disk_map[disk_info['Serial Number']] = disk_info
        myfile.close()
    return disk_map


def get_storagearray_info(arrayinfo_file, disk_file_list):
    rx_array = re.compile(r'''
    ^
    SUMMARY.+\n
    (?P<array_info>[\s\S]+?)
    System\sLocation.+
    (?P<noneed_info1>[\s\S]+)
    Member\sDisk\sInfo.+
    (?P<disk_info>[\s\S]+)
    HyperMetro\sSummary.+
    (?P<noneed_info>[\s\S]+)
''', re.MULTILINE | re.VERBOSE)

    rx_disk_in_array = re.compile(r'''
    ^
    .*Disk\sID.+\n
    (?P<diskinfo>[\s\S]+?)
    Disk\sElabel.+\n
    (?P<noneed_info2>[\s\S]+?)
    (?=^.+Disk\sID|\Z)
''', re.MULTILINE | re.VERBOSE)

    storage_array = dict()
    data = []
    with open(arrayinfo_file, 'r') as myfile:
        data = myfile.read()
    array_match = rx_array.finditer(data)
    for array in array_match:  # should only be one array
                                                            338,5         62%
```



```python
    for array in array_match:  # should only be one array
        for info in array.group('array_info').strip().split("\n"):
            storage_array[info.split(":")[0].strip()] = info.split(":")[1].strip()
        disk_in_array_match = rx_disk_in_array.finditer(array.group('disk_info'))
        for disk in disk_in_array_match:  # this loop is just to get disk type and vendor, will
break on first findings
            temp_disk_info = dict()
            for info in disk.group('diskinfo').strip().split("\n"):
                temp_disk_info[info.split(":")[0].strip()] = info.split(":")[1].strip()
            if "NVMe_SSD" in temp_disk_info['Disk Type'] and ("HUAWEI" in temp_disk_info['Disk V
endor'] or "HUASY" in temp_disk_info['Disk Vendor']):
                disk_map = get_disklist_map_nvme_huawei(disk_file_list)
            elif"SSD" in temp_disk_info['Disk Type'] and ("HUAWEI" in temp_disk_info['Disk Vendo
r'] or "HUASY" in temp_disk_info['Disk Vendor']):
                disk_map = get_disklist_map_sas_huawei(disk_file_list)
            else:
                return dict()  # not the disks we are processing now
            break
        for disk in disk_in_array_match:
            temp_disk_info = dict()
            for info in disk.group('diskinfo').strip().split("\n"):
                temp_disk_info[info.split(":")[0].strip()] = info.split(":")[1].strip()
            update = disk_map.get(temp_disk_info['Disk SN'])
            update['Disk Type'] = temp_disk_info['Disk Type']
            update['Disk Vendor'] = temp_disk_info['Disk Vendor']
            update['Disk Domain ID'] = temp_disk_info['Disk Domain ID']
            disk_map[temp_disk_info['Disk SN']] = update
        storage_array['Member Disk List'] = list(disk_map.values())
    myfile.close()
    return storage_array

def get_collecttime(time_file):
    with open(time_file, 'r') as myfile:
        data = myfile.read()
    for info in data.split("\n"):
        if "fileCreateTime" in info.split("=")[0]:
            collecttime = info.split("=")[1].strip()
            break
    time_stamp = time.strftime('%Y-%m-%d-%H-%M-%S', time.gmtime(int(collecttime)/1000))
    myfile.close()
    return  time_stamp

def process_dorado_data_file(files):
    disk_file_list = []
    for file in files:
        if "Running_Data/config.txt" in file:
            array_config_file = file
        if"disksmartinfo/smart" in file:
            disk_file_list.append(file)
        if"pkginfo/desc" in file:
            timestamp_file = file
    storagearray_info = get_storagearray_info(array_config_file, disk_file_list)
    time_stamp = get_collecttime(timestamp_file)
    storagearray_info['timestamp'] = time_stamp
    return storagearray_info


def unpackAndPost(filename,filepath):
    target_dir = filepath.replace('.','_')
    fmeta = (filename.split('.')[0]).split('_')
    print("[unpackAndPost] file meta:{}".format(fmeta))
    if len(fmeta) < 4:
        print("[unpackAndPost] Error: unable to collect all DC meta data from the filename:{}".f
ormat(filename))
        if "Data_Dorado" in filename:
            print("[unpackAndPost] try unpackAndPostDoradoData".format(filename))
            unpackAndPostDoradoData(filename, filepath)
        return
    print("[unpackAndPost] dc file={}".format(filename))
    system_id = fmeta[0]
    dc_type = fmeta[1]
    collect_id = fmeta[2]
    timestamp = fmeta[3]

    try:
        #clean if the target_dir already exists
        if os.path.exists(target_dir):
            shutil.rmtree(target_dir)
    except:
        print("exception thrown while cleaning up old downloaded file")
                                                            414,5         80%
```



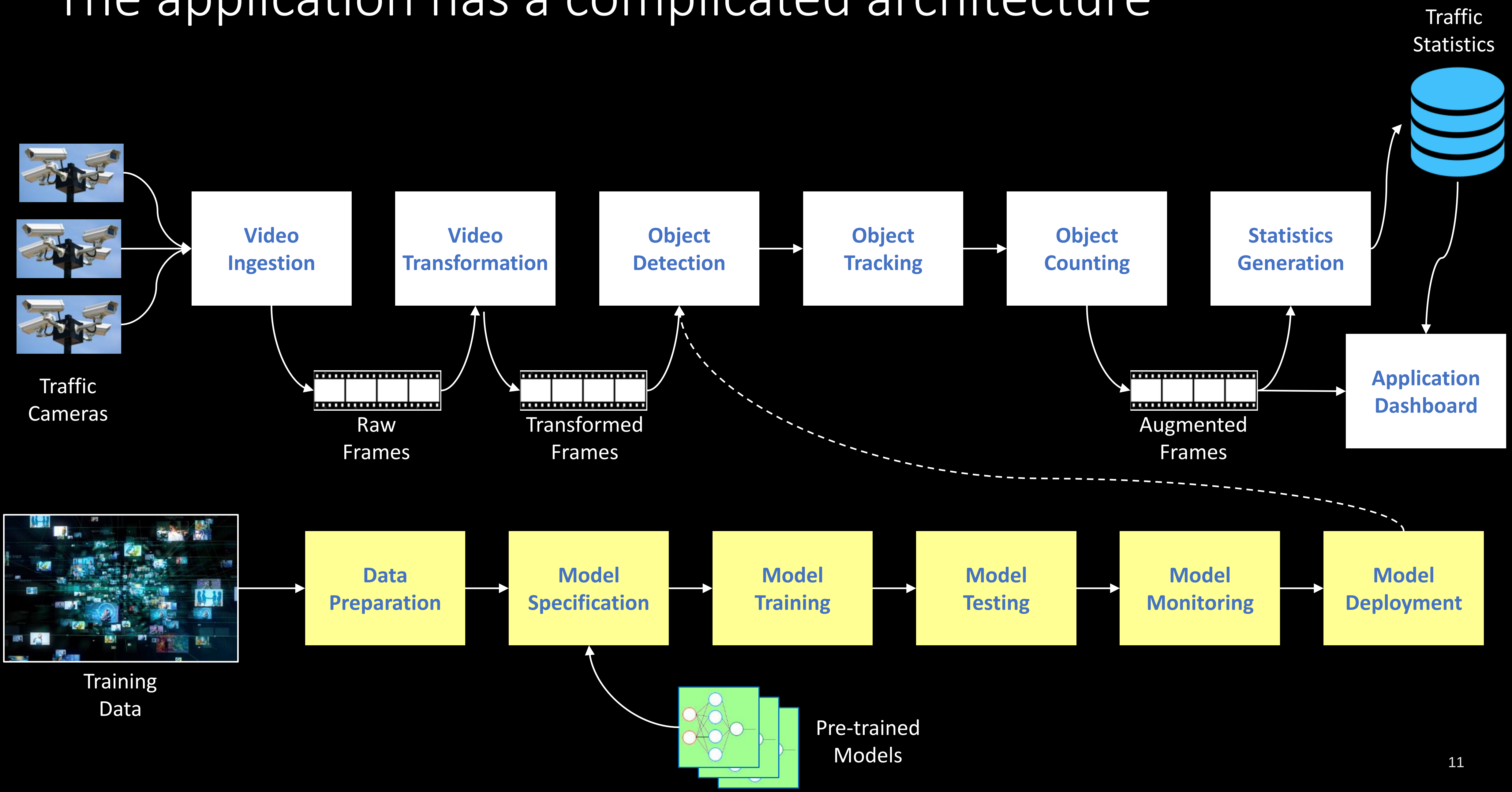
```python
#        script = ["python2.7", "run.py"]
#        process = subprocess.call(" ".join(script), shell=True, env={"PYTHONPATH": "."}, cwd="O
bsCmd")
#    except:
#        print("Didn't upload files to Archive !!")

    cmd = "mkdir " + target_dir

    print("[unpackAndPost]{}".format(cmd))
    os.system(cmd)

    tar = tarfile.open(filepath)
    tar.extractall(path=target_dir)

    try:
        #remove downloaded file
        if os.path.exists(filepath):
            os.remove(filepath)
    except:
        print("exception thrown while removing {}".format(filepath))
        return
    try:
        if os.path.exists(target_dir):
            files = findFilesFromDC(target_dir)
        for file in files:
            print(file)
            cacheAndPostMQ(system_id, dc_type, collect_id, timestamp, file)
    except:
        print("exception thrown while crawling package:{}".format(target_dir))
        return

    try:
        #clean uploaded files
        if os.path.exists(target_dir):
            shutil.rmtree(target_dir)
    except:
        print ("exception thrown while cleaning up downloaded file:{}".format(target_dir))
        return

    print("[unpackAndPost] completed posting new dc:{} to BDL listener.".format(filename))

@app.route('/')
def version():
    return APP_NAME + " v" + APP_VERSION

@app.route('/post_form')
@auth.login_required
def post_form():
    return render_template('upload.html')

@app.route('/upload', methods=['GET', 'POST'])
@auth.login_required
def upload():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit a empty part without filename
        if file.filename == '':
            return "No selected file"
        if file and is_allowed_file(file.filename):
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)
            #unpackAndPost(filepath)
            future = threadPool.submit(unpackAndPost, filename, filepath)
            return "File upload is complete:" + filename
    return "No file post made"

@auth.verify_password
def verify(username, password):
    if not (username and password):
        return False
    return USER_DATA.get(username) == password

if __name__ == '__main__':
    #app.run(processes=3)
    app.run(debug=True,host='0.0.0.0',port=int(APP_PORT))
                                                            501,5         Bot
```

A Demonstration

Let us use Wenju to build a traffic surveillance application

- Ingest videos from traffic cameras at difficult locations
- Build ML models for detecting and tracking vehicles and people in the videos
- Analyze traffic in real time
- Report historical and current traffic conditions

The application has a complicated architecture



A Difficult Job Without Wenju

- Requiring the mastery of many tools

- as well as in-depth knowledge on
  - Video analytics
  - Data analytics
  - Computer vision algorithms
  - Model training and tuning
  - Data ops, model ops, and SaaS ops
  - Data and model lineage
  - Enterprise-grade performance and scalability
  - Infrastructure management and optimization

Wenju makes it a lot simpler and faster to develop this application

Check out the demo video to see how

The Benefits of Wenju

- Alignment of AI with business processes
  - Drive AI with a wide spectrum of business data
  - Feed business operations with deep insight in real time
- Democratization of AI for enterprises of all sizes
  - Reduce the level of expertise needed through abstraction and reuse
  - Enable different personas to be self-sustained
- Fast delivery of true value to the enterprise
  - Allow teams to collaborate effectively across organizational boundaries
  - Streamline and simplify lifecycle management of all application components
- Compliance with regulations and enterprise policies
  - Maintain the traceability of datasets and models
  - Ensure data privacy and model trustworthiness