

Byzer

Everything is a table.

An open-source programming language for
Data & AI

William Zhu

2021.12

Agenda

- ◆ 落地 Data & AI 的痛点
- ◆ Byzer 语言介绍
- ◆ Byzer 语言技术原理与能力展示
- ◆ Byzer 生产案例
- ◆ 预告

Kyligence 公司介绍

自主开源技术，打造开源生态



- ◆ 全球领先的大数据 OLAP 领导者
- ◆ 中国首个 Apache 顶级开源项目
- ◆ 1500+ 全球生产用户



- ◆ 面向 Data + AI 的类 SQL 语言
- ◆ AI + Big Data 领域开源新秀
- ◆ 金融、互联网等行业应用案例

SQL为王，为数据分析师、数据科学家、数据工程师提供统一的全栈平台

企业落地 Data + AI 所面临的痛点

- ◆ ROI 低

企业发现落地一个算法到具体的某个场景的成本远高于其带来的收益



- ◆ 缺少ML专家

企业愿意花钱，但依然难以构建完整平台和招募到足够的研发和数据科学家



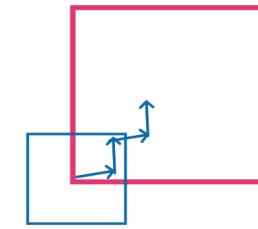
- ◆ 机器学习实操困难

模型和场景难结合
需要大量的探索和测试



- ◆ 部署困难

复杂性以适应生产规模
难以维护部署，工作流程重复



数据工程师落地 Data + AI 所面临的痛点

数据处理链路过长

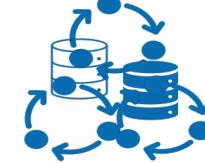
1. 使用SQL做一些数据预处理



6. 使用调度连接组件任务



5. 模型迭代



2. PySpark进一步处理数据



3. 使用机器学习库进行训练



4. 复杂的模型部署



数据工程师落地 Data + AI 所面临的痛点

数据流转、管控、运维复杂

- ◆ 数据需要在各个组件流转
- ◆ 数据格式，形态都不一致
- ◆ 多次落盘，产生大量临时数据



- ◆ 各个地方都需要权限管控
- ◆ 容易产生漏洞，且管理繁琐
- ◆ 各系统的能力不一致
- ◆ 使用者需要学习各个系统
- ◆ 研发需掌握各类技术栈
- ◆ 培训、交接困难

如何真正提高效率来落地数据平台和 AI 工程化?

- ◆ 开发一个更强大易用的框架?
- ◆ 依托公有云提供的托管服务?
- ◆ 招聘高水平的研发团队?



我们认为提高效率，一定要从**编程语言层面**进行革新

- ◆ 声明式融合命令式语言，易于上手又不失灵活
- ◆ 云原生设计，充分利用云上算力和便宜的存储
- ◆ 一套语言，一个引擎覆盖数据处理和机器学习，开箱即用

Agenda

- ◆ 落地 Data & AI 的痛点
- ◆ Byzer 语言介绍
- ◆ Byzer 语言技术原理与能力展示
- ◆ Byzer 生产案例
- ◆ 预告

Byzer (Former MLSQL)

A low-code open-source programming language for data pipeline, analytics and AI

- ◆ **Everything is a table**
- ◆ SQL-Like Language
- ◆ Built-In Algorithms And Plugins
- ◆ Customizable, Simple And Powerful
- ◆ Distributed Execution Engine

MLSQL

Make data analysis simple and smart



Byzer

Everything is a table.

<https://github.com/byzer-org/byzer-lang>

Byzer 几大特性

- ◆ 多数据源无缝对接
- ◆ 虚拟表串联数据流
- ◆ 支持模型的批，流，
API的一键部署

万物皆表

统一

- ◆ 对底层数据源无侵入性
- ◆ 支持语言层语法权限控制
- ◆ 支持表，行列级别权限控制
- ◆ 表级别权限可秒级校验

数据安全

高度可扩展

- ◆ 统一语言
 - ◆ 统一权限
 - ◆ 统一资源
 - ◆ 统一管理
-
- ◆ 插件内核（一切皆可插件）
 - ◆ 标准库支持
 - ◆ 支持Python
 - ◆ 支持使用自定义UDF（Scala/Java）

Byzer 常用语法介绍

Byzer

Everything is a table.

- ◆ 数据加载 / Load
- ◆ 数据转换 / Select
- ◆ 保存数据 / Save
- ◆ AI 扩展 / Train | Run | Predict
- ◆ 注册函数，模型 / Register
- ◆ 变量设置 / Set
- ◆ 宏函数 / !Macro Function
- ◆ 代码引入 / Include
- ◆ SQL 分支语句 / If | Else

Byzer 常用语法介绍

- 数据加载 / Load

```
数据格式/名称          路径或者库表  
load excel.`./example-data/excel/hello_world.xlsx`  
where header="true"  
as hello_world;  
表名
```

- 数据转换 / Select

```
load excel.`./example-data/excel/hello_world.xlsx`  
where header="true"  
as hello_world;  
表名  
select hello from hello_world as output;  
原表      新表
```

Byzer 常用语法介绍

- 数据保存 / Save

```
load excel.`./example-data/excel/hello_world.xlsx`  
where header="true"  
as hello_world;  
                                表名  
save overwrite hello_world as csv.`./tmp/hw` where header="true";  
                                路径或者库表  
                                原表    数据格式/名称
```

- 变量设置 / Set

文本类型变量

```
set hello="world";
```

select "hello \${hello}" as title
as output; 变量在 Select 语句中的使用

Byzer 常用语法介绍

- 宏函数 / Macro Function

```
    函数名
set loadExcel = ''
load excel.`{0}`
where header="true"
as {1}
''';
! 调用函数           位置参数1           位置参数2
!loadExcel ./example-data/excel/hello_world.xlsx helloTable;
```

```
    函数名
set loadExcel = ''
load excel.`${path}`
where header="true"
as ${tableName}
''';
! 调用函数           命名参数1           命名参数2
!loadExcel -path ./example-data/excel/hello_world.xlsx -tableName helloTable;
```

Byzer 常用语法介绍

- 分支 / !If | !else

```
select 1 as a as mockTable;
set b_count=`select count(*) from mockTable ` where type="sql" and mode="runtime";
SQL 类型变量
!if ''':b_count == 1 '''; 宏函数，条件表达式
    select 1 as a  as final_table;
!else;
    select 2 as a  as final_table;
!fi;

select * from final_table as output;
```

Byzer 常用语法介绍

- 注册函数，模型 / Register

```
    函数名
register ScriptUDF.`` as arrayLast
where lang="scala"
and code='''def apply(a:Seq[String])={
    a.last  Scala / Java 代码
}'''
and udfType="udf"; UDF 类型: udf / udaf

select arrayLast(array("a","b")) as lastChar as output;

    Select 句式中直接使用
```

Byzer 常用语法介绍

- 代码复用 / Include

```
include project.`./src/common/PyHeader.msql`;
```

项目内脚本地址

项目内引用

第三方库地址

```
include lib.`github.com/allwefantasy/lib-core`  
where  
libMirror="gitee.com"  
and -- commit="xxxxx"  
and alias="libCore";
```

库别名

包引入(hello 函数)

```
include local.`libCore.udf.hello`;  
select hello() as name as output;
```

libCore 库的 hello 函数

引用第三方库

Byzer 常用语法介绍

- 扩展 / Train | Run | Predict

训练数据集

```
train mock_data as RandomForest.`/tmp/models/randomforest` where  
    _____  
        指定算法  
keepVersion="true"  
  
    _____  
        模型保存目录  
  
and evaluateTable="mock_data_validate"  
  
and `fitParam.0.labelCol`="label"  
and `fitParam.0.featuresCol`="features"  
and `fitParam.0.maxDepth`="2";
```

训练

模型转函数

```
register RandomForest.`/tmp/models/randomforest` as model_predict;  
select vec_array(model_predict(features)) as predicted_value from mock_data as output;  
_____  
    算法应用于 Select句式中
```

预测

Byzer 常用语法介绍

- Python 支持

```
run command as Ray.`` where
inputTable="mockTable"      输入表
and outputTable="newMockTable"  结果表
and code='''
from pyjava.api.mlsql import RayContext

ray_context = RayContext.connect(globals(),None)

newrows = []                  获取输入表数据
for row in ray_context.collect():
    row[ "a" ] = 2
    newrows.append(row)

context.build_result(newrows)   构建输出表
''';
使用Python输出
select * from newMockTable as output;
```

Byzer 常用语法介绍

- 插件 / Extension

```
!plugin app remove "mlsql-mllib-3.0";
!plugin app add - "mlsql-mllib-3.0";
```

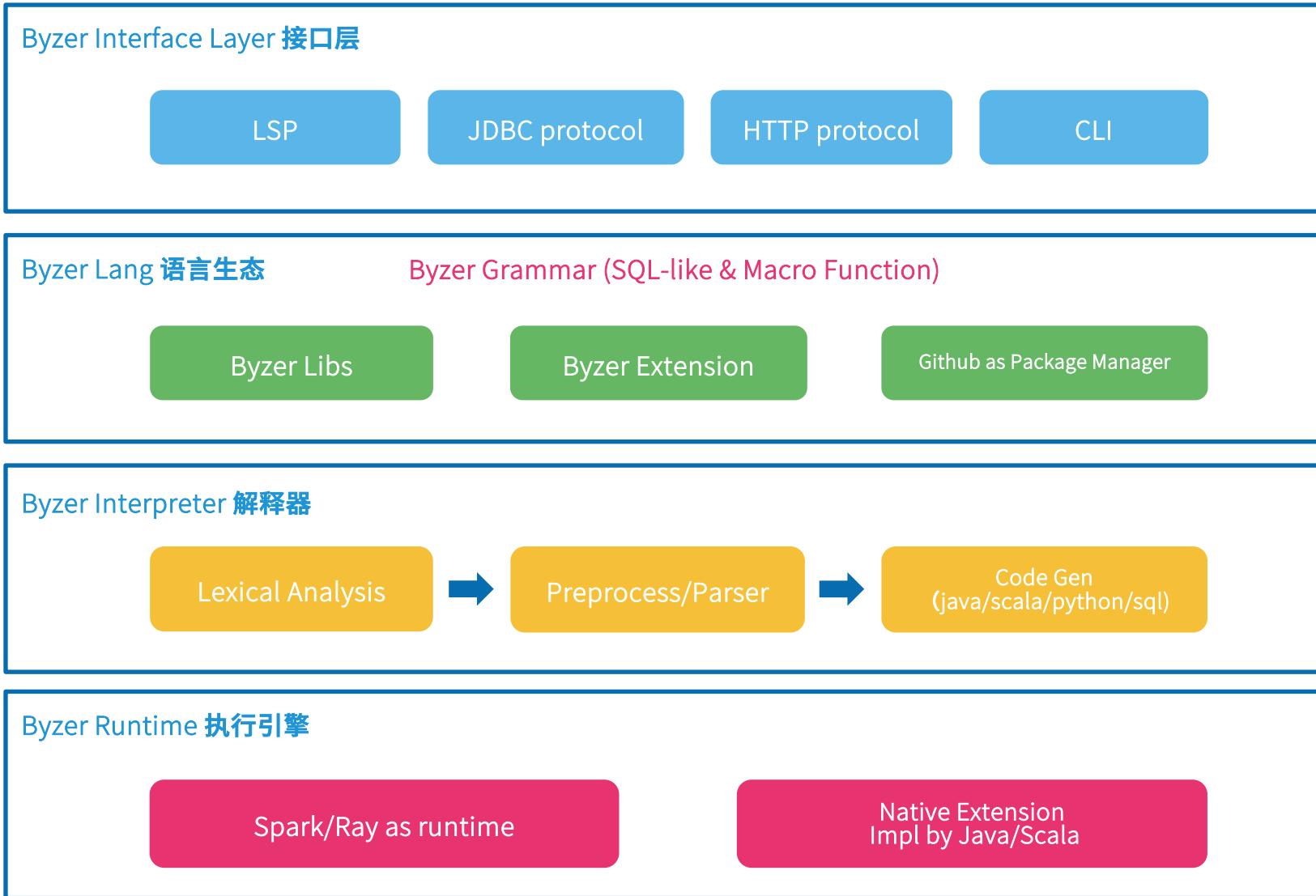
插件名称

```
run command as SampleDatasetExt.`` 随装随用
where columns="id,features,label"
and size="100000"
and featuresSize="100"
and labelSize="2"
as mockData;
```

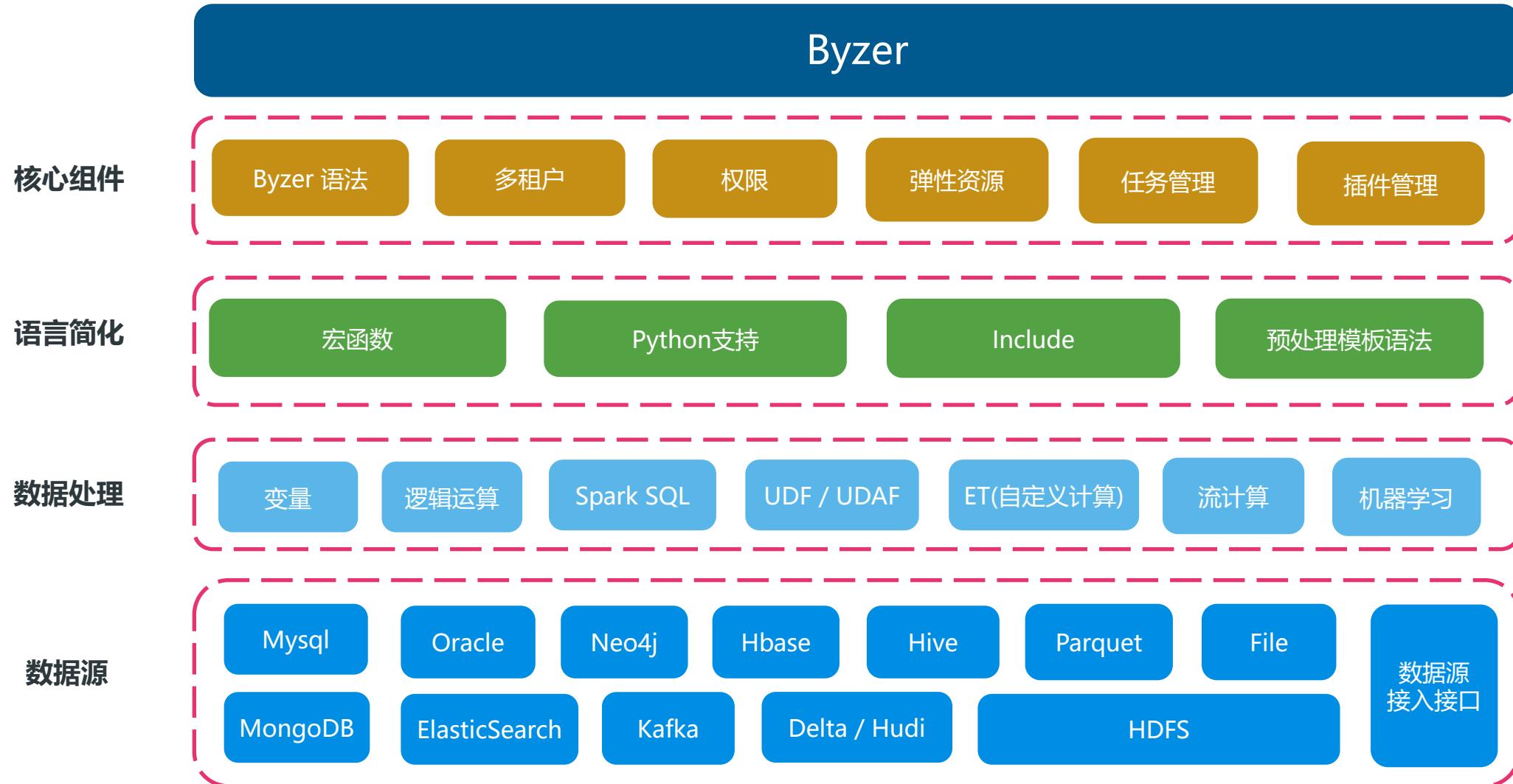
Agenda

- ◆ 落地 Data & AI 的痛点
- ◆ Byzer 语言介绍
- ◆ Byzer 语言技术原理与能力展示
- ◆ Byzer 生产案例
- ◆ 预告

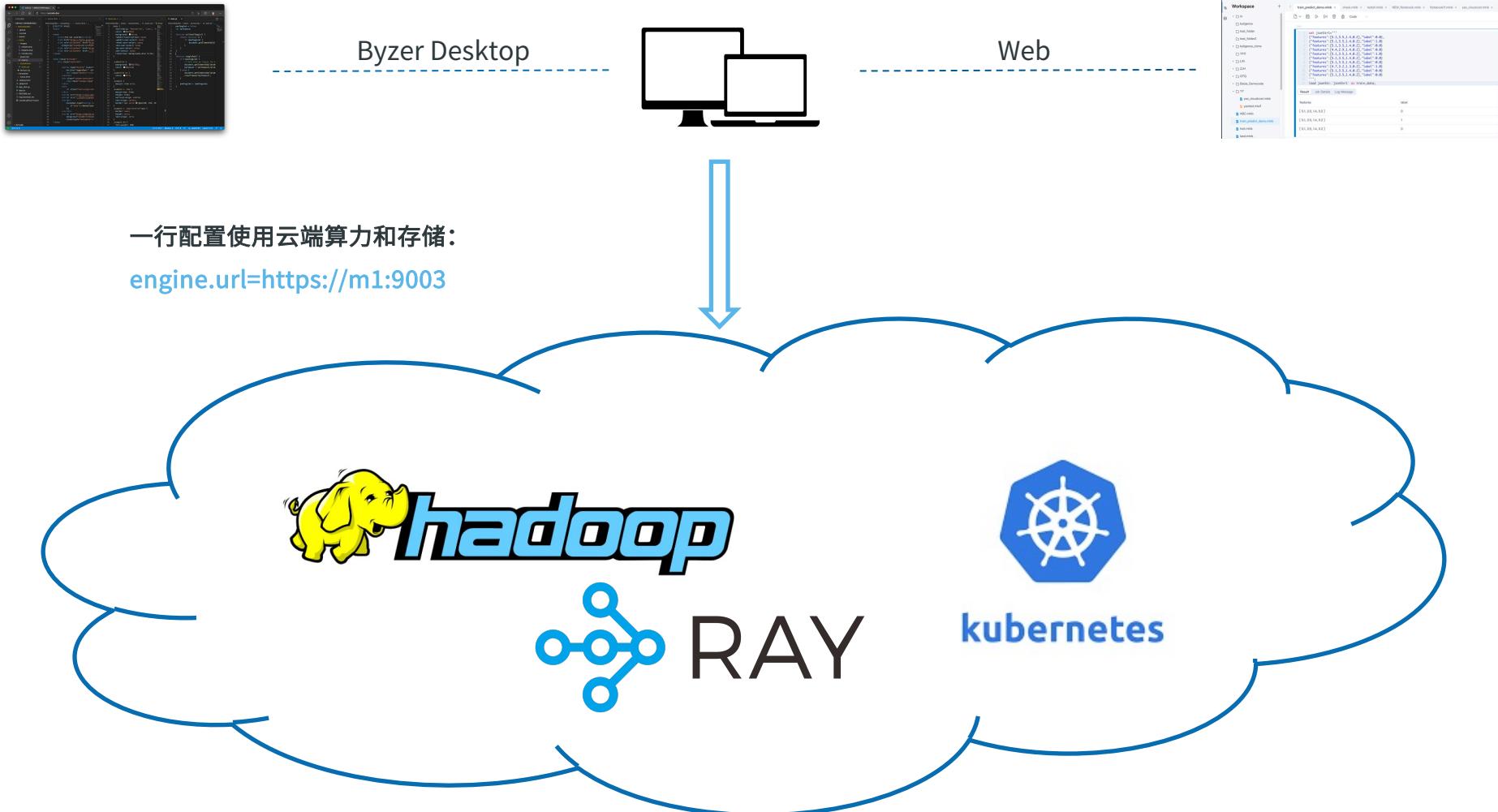
Byzer Architecture



Byzer -功能架构



云原生设计，轻松解锁算力限制



Byzer 能力展示

内置 ET 组件完成 ETL + 特征工程能力

Cell 12

```
1 train data6 as RateSampler.``  
2 where labelCol="label"  
3 and sampleRate="0.7,0.3" as marked_dataset;
```

Cell 13

```
1 select * from marked_dataset where __split__=1  
2 as testingTable;  
3  
4 select * from marked_dataset where __split__=0  
5 as trainingTable;  
6
```

- 内置大量 SQL 特征函数，如 vec_dense 等
- 提供了很多 ET 插件

Cell 14

```
1 --可以用宏命令查看一下数据类型  
2 !desc trainingTable;
```

Byzer 能力展示

利用 Byzer 进行分布式机器学习训练

- AutoML 支持

```
1 train trainData as AutoMLExt.`/tmp/auto_ml_model` where
2   algos="GBTs,LinearRegression,LogisticRegression,NaiveBayes,RandomForest"
3   and keepVersion="true"
4   and evaluateTable="testData";
[17] ✓ 51.6s
...
name          value
-----
modelPath      /_model_1/model/0
algIndex       0
alg            org.apache.spark.ml.regression.LinearRegression
metrics        f1: 0.0 weightedPrecision: 0.0 weightedRecall: 0.0 accuracy: 0.0
status         success
```

Byzer 能力展示

一站式模型发布与部署

```
1 --%deployModel  
2 --%url=http://127.0.0.1:9003  
3 --%access_token=xxxx  
4 -- 把模型部署到指定的服务上  
5 register RandomForest.`/models/rf_model` as model_predict;
```



Result Job Details

function

!

Byzer 能力展示

访问 Rest 接口验证模型

```
1 -- 通过Http请求访问部署后的模型
2 select crawler_http("http://127.0.0.1:9003/model/predict","POST",map(
3 "dataType","row",
4 "sql","select vec_argmax(model_predict(vec_dense(doc))) as predicted_label",
5 "data",'[{"doc":[5.1,3.5,1.4,0.2]}]')
6 )) as res as output;
```

Result Job Details

res

[{"predicted_label":0}]

Total 1 10/page < 1 > Go to 1

↓

Byzer-python 实现 Data & AI 的融合

特征工程

1. Byzer-python 可以无缝获取 Byzer 中的任何视图
2. Byzer-python 处理完后，重新输出成表供后续 Byzer 代码使用
3. 无中间存储
4. Byzer-python 可以实现 GPU 硬件感知，实现环境隔离

```
src > analysis > example > cifar10 > test.kolo
1 !sh wget "https://github.com/allwefantasy/spark-deep-learning-toy/releases/download/v0.01/cifar.tgz";
2 !sh mkdir -p /tmp/cifar10;
3 !sh tar -xf "cifar.tgz" "-C" "/tmp/cifar10";
4
5 load binaryFile.`/tmp/cifar10/cifar/train/*.png` as cifar10; 加载图片
6
7 run command as Ray.`` where
8 inputTable="cifar10"
9 and outputTable="cifar10_resize"      使用 open-cv2 处理图片
10 and code=''''
11 import io, cv2, numpy as np
12 from pyjava.api.mlsql import RayContext
13
14 ray_context = RayContext.connect(globals(),"127.0.0.1:10001")
15
16 def resize_image(row):
17     new_row = {}
18     image_bin = row["content"]
19     oriimg = cv2.imdecode(np.frombuffer(io.BytesIO(image_bin).getbuffer(),np.uint8),1)
20     newimage = cv2.resize(oriimg,(28,28))
21     is_success, buffer = cv2.imencode(".png", newimage)
22     io_buf = io.BytesIO(buffer)
23     new_row["content"] = io_buf.getvalue()
24     new_row["path"] = row["path"]
25     return new_row
26
27 ray_context.foreach(resize_image)
28 '';
29
30 -- 引入一些函数，方便在SQL中使用
31 include project.`./src/common/functions.mlsql`;
32
33 -- 这里，arrayLast 函数来源于 ./src/common/functions.mlsql 中
34
35 select arrayLast(split(path,"/")) as fileName,content      抽取路径中的分类信息
36 from cifar10_resize
37 as final_dataset;
38
39 save overwrite final_dataset as image.`/tmp/size-28x28`    重新保存为图片而不是表
40 where imageColumn="content"
41 and fileName="fileName";
```

Byzer-python 实现 Data & AI 的融合

模型训练

通过 Byzer-python 读取任意 Byzer 表

通过 Byzer-python 返回 [文件目录] 表

通过 Byzer 把模型保存到数据湖

```
70 def epoch_train(weights):
71     sum_weights = reduce(lambda a,b: [(a1 + b1) for a1,b1 in zip(a,b)],weights)
72     averaged_weights = [layer/replica_num for layer in sum_weights]
73     ray.get([worker.set_weights.remote(averaged_weights) for worker in workers])
74     ray.get([worker.train.remote() for worker in workers])
75     return ray.get([worker.get_weights.remote() for worker in workers])
76
77 for epoch in range(6):
78     _weights = epoch_train(_weights)
79
80 model_binary = ray.get(workers[0].get_final_model.remote())
81 [worker.shutdown.remote() for worker in workers]
82 ray_context.build_result(model_binary)
```

files
keras_metadata.pb

saved_model.pb

variables/variables.data-00000-of-00001

variables/variables.index

```
1 save overwrite cifar10_model as delta.`ai_model.cifar_model`;
```

Byzer-python 实现 Data & AI 的融合

模型部署

```
6  
7 register Ray.`cifar_model` as model_predict where  
8 maxConcurrency="2"  
9 and debugMode="true"; → Byzer 支持将 Python 模型注册成 UDF  
10  
11 select model_predict(array(byteArrayToUnsignedIntArray(content)))  
12 from final_cifar10 limit 10 as output;  
13
```

UDF 可以直接应用于 批，流 API 中

Agenda

- ◆ 落地 Data & AI 的痛点
- ◆ Byzer 语言介绍
- ◆ Byzer 语言技术原理与能力展示
- ◆ Byzer 生产案例
- ◆ 预告

案例 – 某消费金融公司落地数据平台案例

业务背景

- ◆ 金融行业，主营业务为个人消费贷
- ◆ 数据安全监管严格，安全合规投入大
- ◆ 受政策强影响，业务调整频繁，人员流动快
- ◆ 业务需求量巨大，紧急需求多
- ◆ IT 团队和设施投入少，人员缩编

技术挑战

- ◆ 数据平台架构设计复杂，扩展性差
- ◆ 技术栈混杂，开发迭代慢，上线流程繁琐
- ◆ 权限管控依赖人工配置，易出错
- ◆ 计算资源无法合理分配，高浪费
- ◆ 研发人员流动无法支撑数据平台架构

案例 - 某消费金融公司落地数据平台案例

使用 Byzer 完成大数据中心平台的迁移和升级

- ◆ 使用 Byzer 在语言层面统一技术栈和架构
- ◆ 生产环境稳定运行**三年多**
- ◆ 累计执行的数据处理任务 **700 万次**
- ◆ 单日执行的调度任务超过 **4000 个**
- ◆ 平台服务用户**日活 50+**
- ◆ 只投入了**2人的研发团队**



+ **Byzer**
Everything is a table.

2 人研发团队支撑生产环境的开发和运维

案例 2-厦门某信息公司

Before



After



痛点

- 搞定一个模型需要 2周 +
- 无法满足复杂的数据处理，如异构 Join
- 必须现场服务器开发

收益

- 4人一月支撑30个模型，500+ 任务
- 同时满足数据处理 & 机器学习需求
- Byzer 支持现场、远程开发部署

Agenda

- ◆ 落地 Data & AI 的痛点
- ◆ Byzer 语言介绍
- ◆ Byzer 语言技术原理与能力展示
- ◆ Byzer 生产案例
- ◆ 预告

预告

Byzer Org 白泽社区将在 12.21 (冬至) 正式开源

- ◆ 前身为 MLSQL 社区，核心是 Byzer-lang
- ◆ 面向 Data & AI，围绕 Byzer 打造技术生态
- ◆ 旨在释放分析师、工程师、运维人员的生产力
- ◆ 帮助用户以**低成本、高效率**的方式落地数据平台和完成 AI 工程化
- ◆ <https://github.com/byzer-org>

The screenshot shows the GitHub organization page for 'Byzer Org'. The header includes the organization logo ('Byzer - Everything is a table.'), the name 'Byzer Org', a description 'Everything is a table.', and links for Overview, Repositories (15), Packages, People (23), Teams (10), Projects (1), and Settings. The main content area features a 'README.md' file with the text 'Hey, this is Byzer Community 🙌'. It highlights the 'Byzer' logo and the tagline 'Everything is a table.'. Below this, it states 'Yes, we are building a new opensource community on GitHub which named Byzer' and lists two bullet points: 'Byzer: A mythical creatures in chinese myth system, it can speak in human languages and knows everything' and '白泽: 中国古代神话中的瑞兽。能言语, 通万物之情, 知鬼神之事'. A section titled 'A Data-oriented and AI-oriented community' explains the goal of developing programming language and products for Data Engineers, AI Engineers, and Operators in the domain of Big Data & AI. It also mentions the SMB (Small Medium Business) target audience and lists three contributing projects: 'Byzer-Lang', 'Byzer Notebook', and 'Byzer Docs'. The footer notes that the community is sponsored by Kyligence Inc. and encourages contributions.

联系我们



Kyligence Inc

- ◆ <http://kyligence.io>
- ◆ info@kyligence.io
- ◆ Twitter: @Kyligence



Byzer Org

- ◆ <https://github.com/byzer-org>
- ◆ hailin.zhu@kyligence.io

