

使用Jupyter赋能☁️服务

演讲人：丁来强





使用Jupyter赋能☁️服务

- ▶ 关于Jupyter
- ▶ 扩展Jupyter
- ▶ 云计算与Jupyter
- ▶ 演示



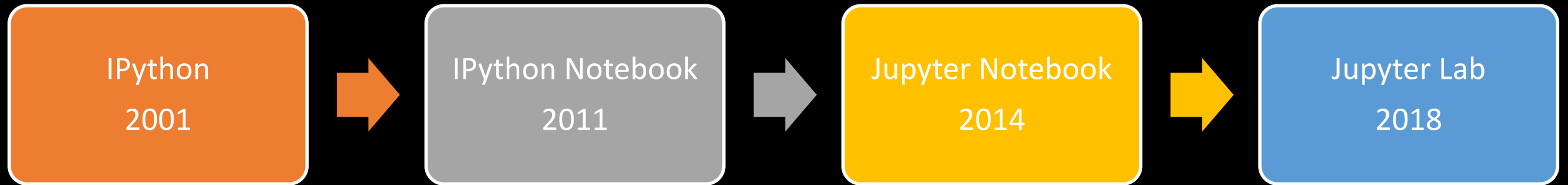
1

关于Jupyter

介绍Jupyter历史、现状、未来趋势等



Jupyter的前世今生





Jupyter的前世今生

IPython
2001



```
1. IPython: settings/20181015 (python3.7) — 87X21
wjo1212deMacBook-Pro:20181015 wjo1212$ clear
wjo1212deMacBook-Pro:20181015 wjo1212$ IPython3
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import sys

In [2]: sys.path
Out[2]:
['',
 '/anaconda3/bin',
 '/anaconda3/lib/python37.zip',
 '/anaconda3/lib/python3.7',
 '/anaconda3/lib/python3.7/lib-dynload',
 '/anaconda3/lib/python3.7/site-packages',
 '/anaconda3/lib/python3.7/site-packages/aeosa',
 '/anaconda3/lib/python3.7/site-packages/IPython/extensions',
 '/Users/wjo1212/.ipython']

In [3]: █
```



Jupyter的前世今生

IPython Notebook
2011



IPython Notebook NotebookEx Last saved: Jul 26 1:06 PM

```
In [1]: import numpy as np
import pandas as pd
print 'Hello world!'

Hello world!

In [2]: tips = pd.read_csv('book_scripts/ch08/tips.csv')
tips.head()

Out[2]:
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
In [4]: img = plt.imread('book_scripts/ch03/stinkbug.png')
figure(figsize=(4, 4))
plt.imshow(img)

Out[4]: <matplotlib.image.AxesImage at 0x7f465d34a510>
```

Jupyter Notebook
2014



jupyter color_scatterplot Last Checkpoint: 11/16/2014 (autosaved)

```
In [4]: import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

In [5]: N = 4000

In [6]: x = np.random.random(size=N) * 100
y = np.random.random(size=N) * 100
radii = np.random.random(size=N) * 1.5
colors = ['#%02x%02x%02x' % (r, g, 150) for r, g in zip(np.floor(50+2*x), np.floor(30+2*y))]

In [7]: output_notebook()

BokehJS successfully loaded.

In [8]: p = figure()
p.circle(x, y, radius=radii, fill_color=colors, fill_alpha=0.6, line_color=None)

Out[8]: <bokeh.plotting.Figure at 0x10a03ccc0>

In [9]: show(p)
```



Jupyter的前世今生

Jupyter Lab
2018



The screenshot displays the Jupyter Lab interface in a browser window at localhost:8888/lab. The interface is divided into several sections:

- Files:** A sidebar on the left showing a file explorer with a list of files and their last modified dates. The file "World Happiness Inde..." is selected.
- Code Editor:** The main area shows a Python notebook cell with the following code:

```
sns.set_context("notebook", font_scale=1.0)
plt.figure(figsize=(15, 10))
sns.boxplot(x="Region", y="Happiness Score", data=df)
plt.xticks(rotation=50)
```

The output (Out[354]) shows a box plot of Happiness Score by Region. Below the plot, the code for a pair grid is shown:

```
In [355]: #Happiness Score & Life Expectancy
g = sns.PairGrid(df1, vars=["Happiness Score", "Life Expectancy"])
g.map(plt.scatter)
```

The output (Out[355]) shows a pair grid of scatter plots for Happiness Score and Life Expectancy.
- Heatmap:** A second notebook cell shows code for a heatmap:

```
In [10]: #Heatmap to find correlation between each of the variables
import matplotlib inline
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
f, ax = plt.subplots(figsize=(10, 8))
corr = df1.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=bool), square=True, ax=ax)
```

The output (Out[10]) shows a heatmap of the correlation matrix for the variables: Happiness Rank, Happiness Score, Lower Confidence Interval, Upper Confidence Interval, Economy (GDP per Capita), Family, Health (Life Expectancy), Freedom, Trust (Government Corruption), Generosity, and Dystopia (Rebellion). The heatmap shows strong positive correlations between Happiness Score and other variables, and strong negative correlations between Dystopia (Rebellion) and other variables.

Jupyter现状：很流行

01



NumPy Bokeh
SciPy. pandas
matplotlib

Python科学生态入口

02



Python R JS
C# Microsoft .NET
Scala

开放生态、50+语言

03



Amazon SageMaker Google Colab
GitHub Azure Notebook
Alicloud PyODPS

云计算界面



交互式计算趋势：Jupyter为啥流行？



强大、正经

交互式计算变得“正经”：工具越来越强大、丰富。也逐渐走向开放标准化、生态丰富。

分析洞察编程

“分析洞察”式编程正在“兴起”：区别于泛化任务式编程。

超越数据共享

共享不再是仅仅数据：还包括计算、结果与叙述等。

1

2

3

4

5

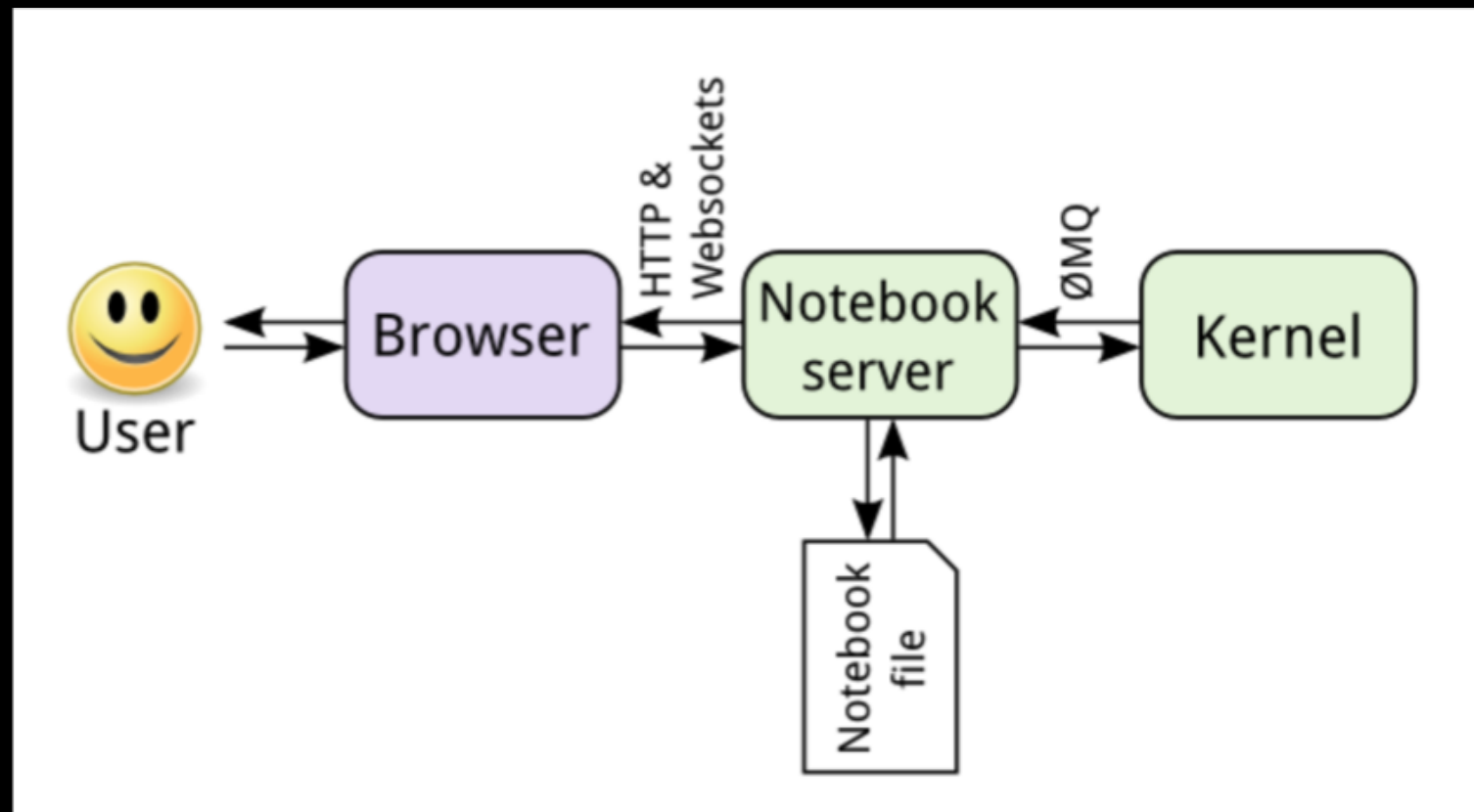
计算性叙述

计算性叙述变得“流行”：代码、文本、可视化融合。

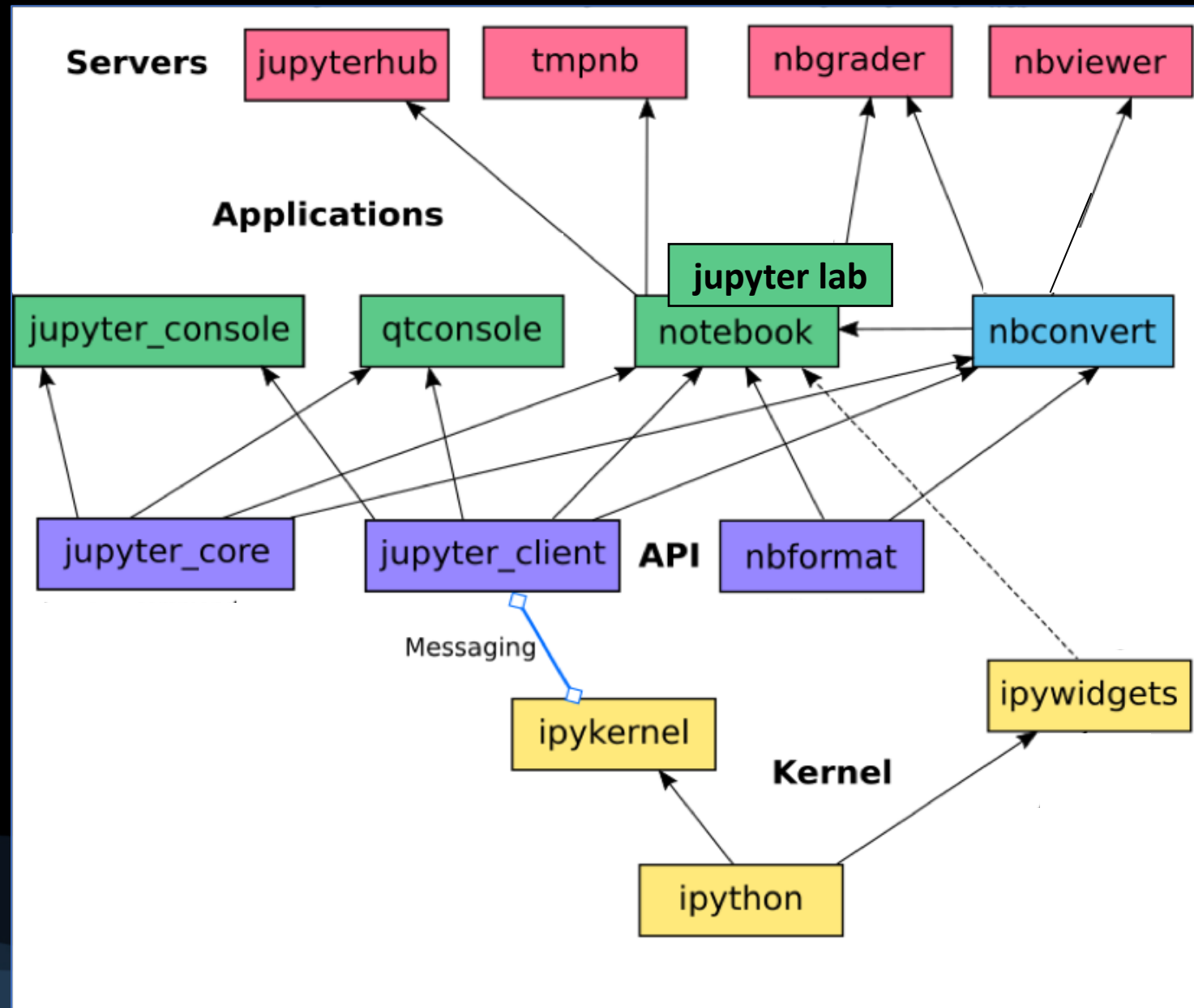
拥抱多语言

个人与组织拥抱多语言：跨语言的统一运行、交互与协作。

Jupyter架构



Jupyter架构





2

扩展Jupyter

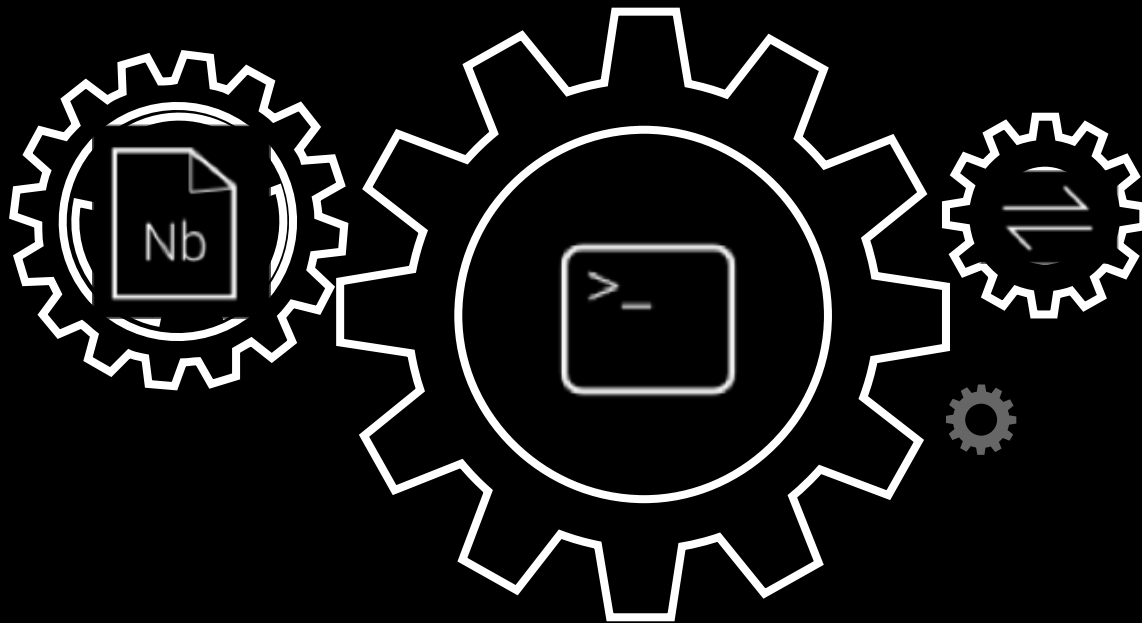
Jupyter如何被扩展？扩展的场景、模式是什么？





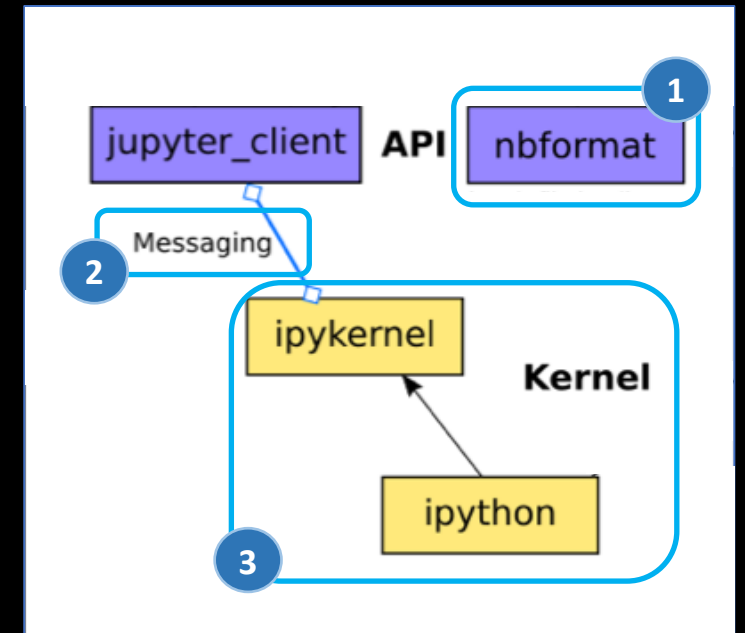
Jupyter的开放标准

1. Notebook
文档格式



2. 交互计算
协议

3. Kernel





Jupyter NB 格式

```
In [28]: %time
for x in range(10):
    print(x)
```

CPU times: user 3 μ s, sys: 1 μ s, total: 4 μ s

Wall time: 10 μ s

0

1

2

3

4

5

```
{
  "cell_type": "code",
  "execution_count": 28,
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "CPU times: user 3  $\mu$ s, sys: 1  $\mu$ s, total: 4  $\mu$ s\n",
        "Wall time: 10  $\mu$ s\n",
        "0\n",
        "1\n",
        "2\n",
        "3\n",
        "4\n",
        "5\n",
        "6\n",
        "7\n",
        "8\n",
        "9\n"
      ]
    }
  ],
  "source": [
    "%time\n",
    "for x in range(10):\n",
    "    print(x)"
  ]
},
```

Jupyter扩展策略

1. 扩展 IPython
Kernel

2. 扩展新Kernel

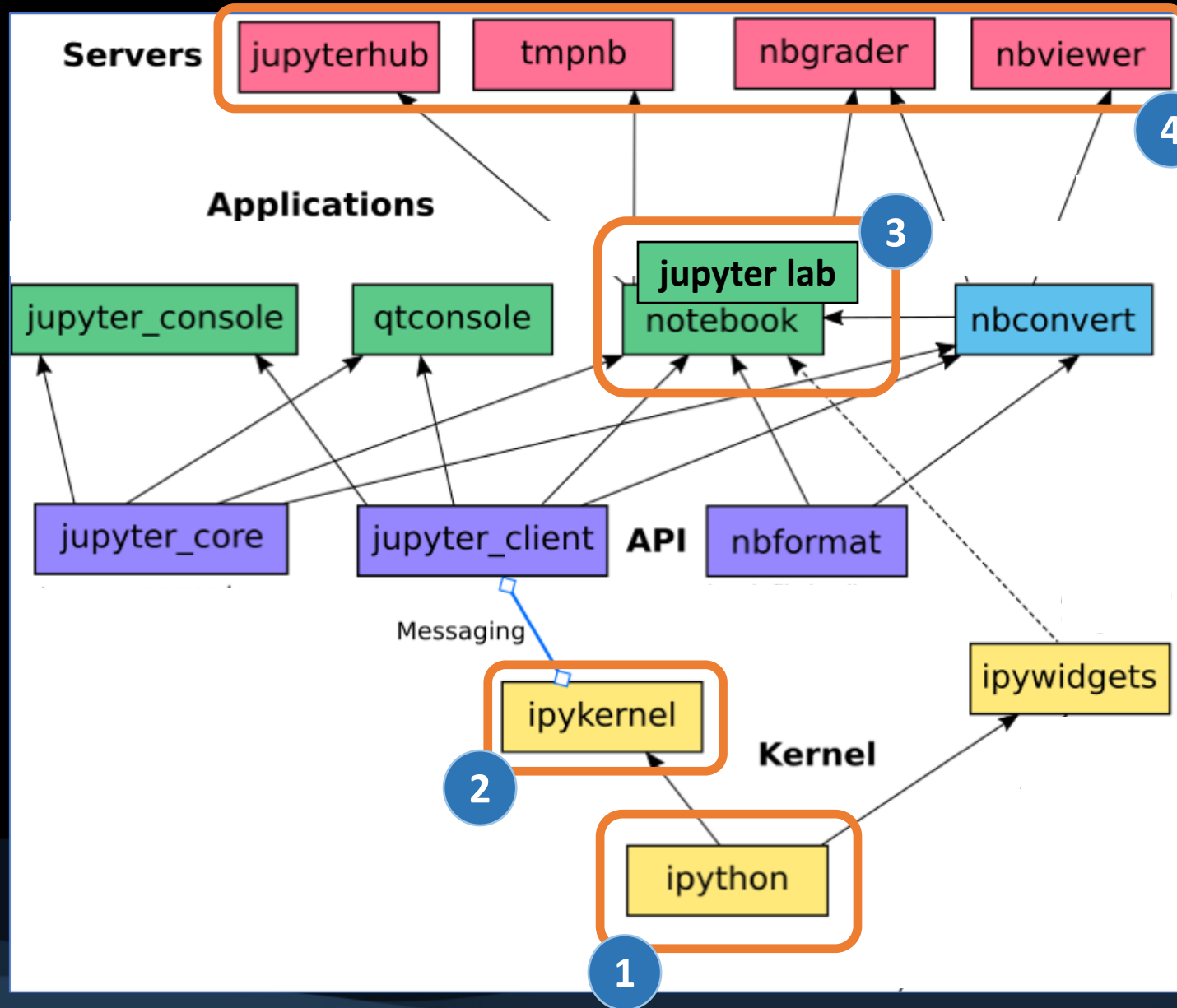
3. 扩展Notebook

4. 扩展架构





Jupyter扩展策略





#1. 扩展 IPython Kernel

- 加载插件
- 扩展 Magic Command
- 扩展输入格式



加载插件

- %load_ext , %unload_ext , %reload_ext

```
In [ ]: %load_ext my_extension
```

my_extension.py

```
def load_ipython_extension(ipython):  
    ipython.register_magic(SLSMagics)  
    sls_client = LogClient(...)  
    ipython.push({'sls', sls_client})  
  
def unload_ipython_extension(ipython):  
    # do some cleanup work
```



扩展 Magic Command

- 如何添加一个魔法方法，执行一些任务？

```
In [11]:
```

```
%manage_log
```

```
Connection to Alicloud Log Service: Verified.  
Created Log Service Client as 'log'  
Created OSS Service Client as 'oss'
```



扩展 Magic Command

```
from IPython.core.magic import (Magics,
                                magics_class, line_cell_magic)

@magics_class
class SLSMagics(Magics):

    @line_cell_magic
    def manage_log(self, line, cell=None):
        ....

    @line_cell_magic
    def log(self, line, cell=None):
        ....

def load_ipython_extension(ipython):
    ipython.register_magic(SLSMagics)
    log_client = LogClient(...)
    ipython.push({'log', log_client})
```

In [11]: %manage_log

```
Connection to Alicloud Log Service: Verified.
Created Log Service Client as 'log'
Created OSS Service Client as 'oss'
```



Magic Command 支持库

```
In [8]: %log
population > 1000
| select province as "省份", count(1) as "订单" group by province desc
```

Out[8]:

	省份	订单
0	上海	10000
1	江苏	2000
2	山东	4900

```
# 函数标记
from IPython.core.magic import line_cell_magic, needs_local_scope, line_magic

# 类标记
from IPython.core.magic import magics_class

# 命令参数支持
from IPython.core.magic_arguments import argument, magic_arguments
```

扩展输入格式

- 如何自动将Python2代码转换为Python3 ?



The screenshot shows a Jupyter Notebook interface. The top menu bar includes '文件', '编辑', '查看', '插入', '单元格', '服务', 'Navigate', '可信的', and 'Python 3'. Below the menu is a toolbar with icons for '+', '剪贴板', '运行', '代码', and 'Notify: Disabled'. The main area shows a code cell with the input 'In [3]: print "hello world"' and the output 'hello world'. The 'Python 3' dropdown menu is highlighted with a blue dashed border.

```
In [3]: print "hello world"

hello world
```



扩展输入格式

- 如何自动将Python2代码转换为Python3 ?

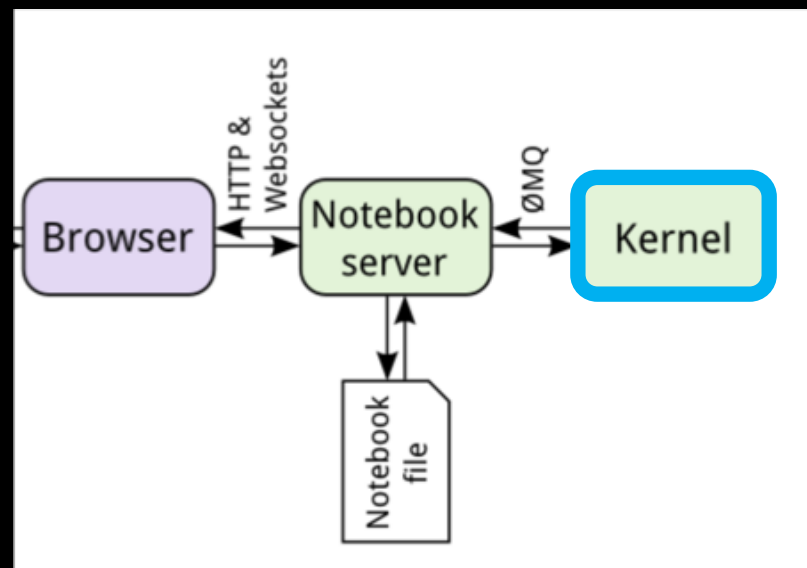
```
In [ ]: %load_ext my_extension
```

my_extension.py

```
def auto2to3(lines):  
    code = ''.join(lines)  
    return autoformat_2to3(code)  
  
def load_ipython_extension(ipython):  
    ipython.input_transformers_cleanup.append(auto2to3)
```

#2. 扩展新 Kernel

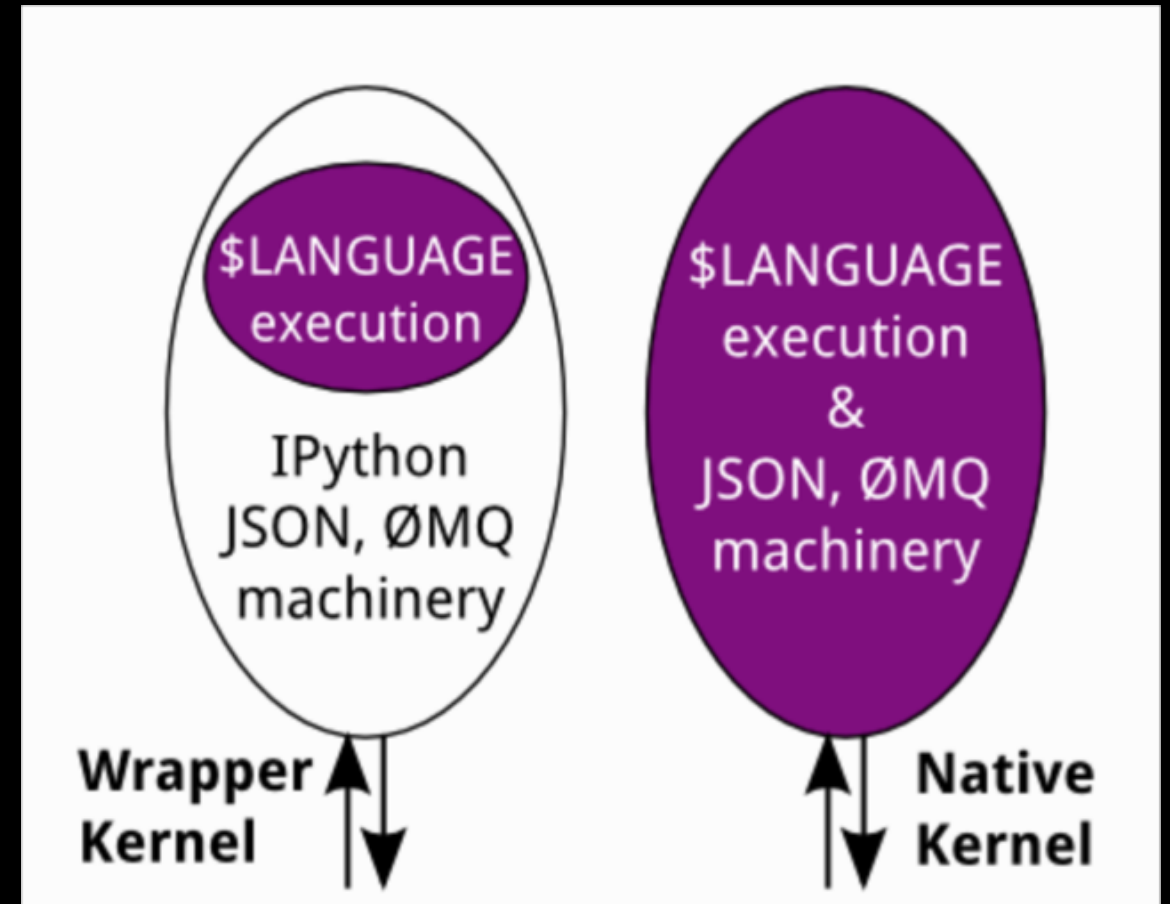
- 场景：
 - 支持一种新的语言
 - 让这个语言支持交互（例如C++）
 - 支持一种新的DSL
 - 也可能是多个语言的混合或者子集
 - 提供一种新的执行环境
 - 环境可能在Docker、远程、云端等





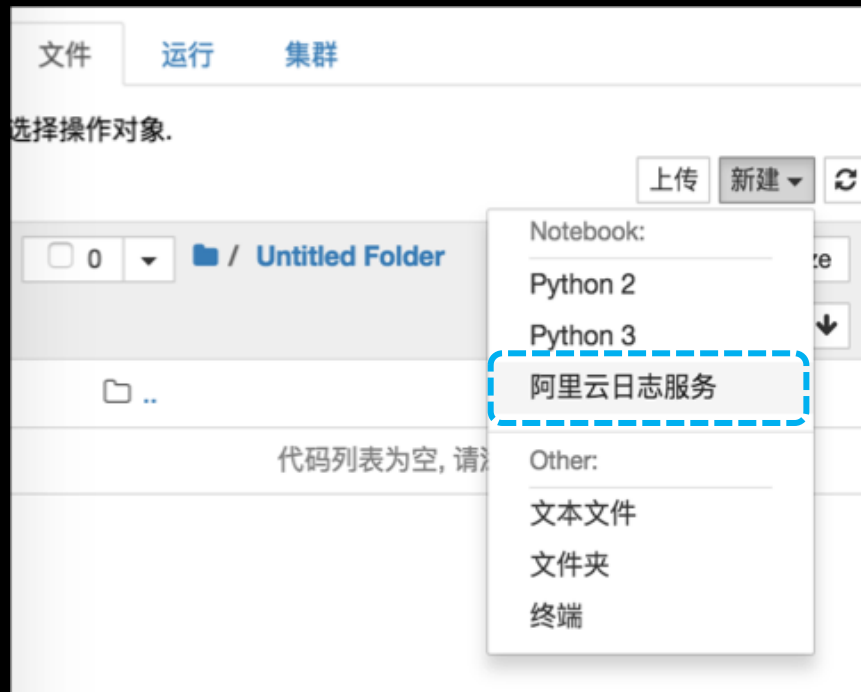
实现策略

- 1. 复用Wrapper Kernel
 - 只需实现语言执行部分
 - 简单高效
- 2. 重头实现
 - 需要重现通讯与执行部分
 - 复杂强大





创建一个Kernel



jupyter log service (未保存改变)

文件 编辑 查看 插入 单元格 服务 Widgets 帮助 可信的 阿里云日志服务

运行

In [4]: `!configure`

账户 高级

默认区域: 上海

密钥ID: ABCD12314

密钥Key: QWERTASAF

确定

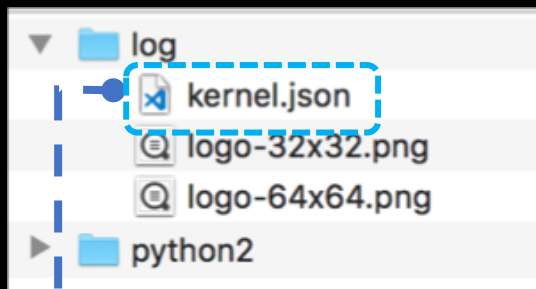
In [8]: `population > 1000`
`| SELECT province AS "省份", COUNT(1) AS "订单" GROUP BY province DESC`

Out[8]:

	省份	订单
0	上海	10000
1	江苏	2000
2	山东	4900

创建一个Kernel

/usr/local/share/jupyter/kernels/



kernel.json

```
{  
  "argv": [  
    "python",  
    "-m",  
    "alicloud_log",  
    "-f",  
    "{connection_file}"  
  ],  
  "display_name": "阿里云日志服务"  
}
```

alicloud_log.py

```
from ipykernel.kernelbase import Kernel  
  
class LogKernel(Kernel):  
    language_info = {  
        'name': 'sql', 'mimetype': 'text/x-sql',  
        'codemirror_mode': 'text/x-sql',  
        'pygments_lexer': 'sql'  
    }  
  
    def do_execute(self, code, silent, store_history=True,  
                  user_expressions=None, allow_stdin=False):  
        result = log.get_data(code)  
  
        self.send_response(self.iopub_socket, 'stream', result)  
  
        return {'status': 'ok',  
                'execution_count': self.execution_count,  
                'payload': [], 'user_expressions': {}, }  
  
if __name__ == '__main__':  
    from ipykernel.kernelapp import IPKernelApp  
    IPKernelApp.launch_instance(kernel_class=LogKernel)
```

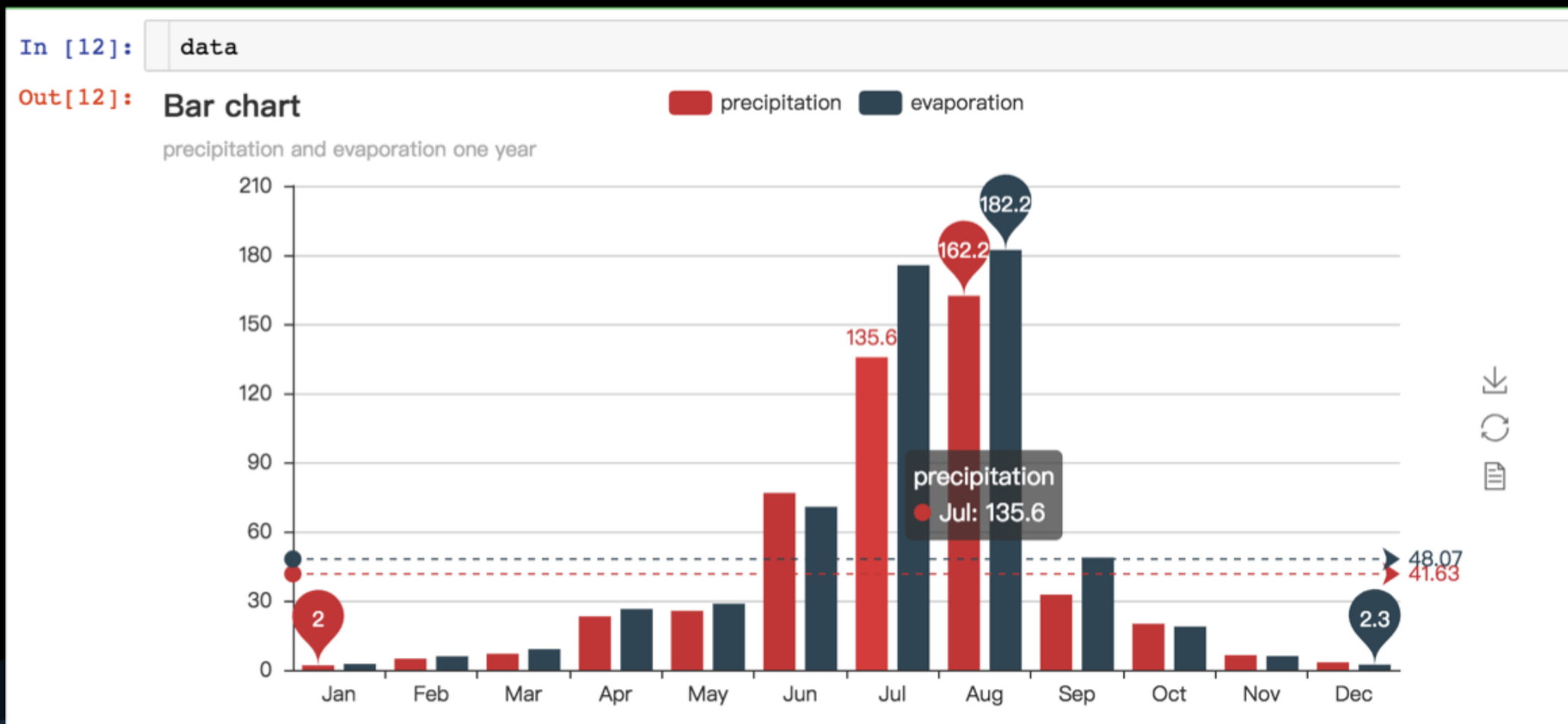
#3. 扩展 Notebook

- 对象自定义可视化
 - 如何将一个对象展示为一个图片、图表、交互图？
- 自定义命令
 - 如何添加一个命令？
- 插件管理
 - 构建NB插件
 - NB插件库



对象自定义可视化

- 如何将一个对象展示为一个图片、图表、交互图？





对象自定义可视化

- data._repr_html_()

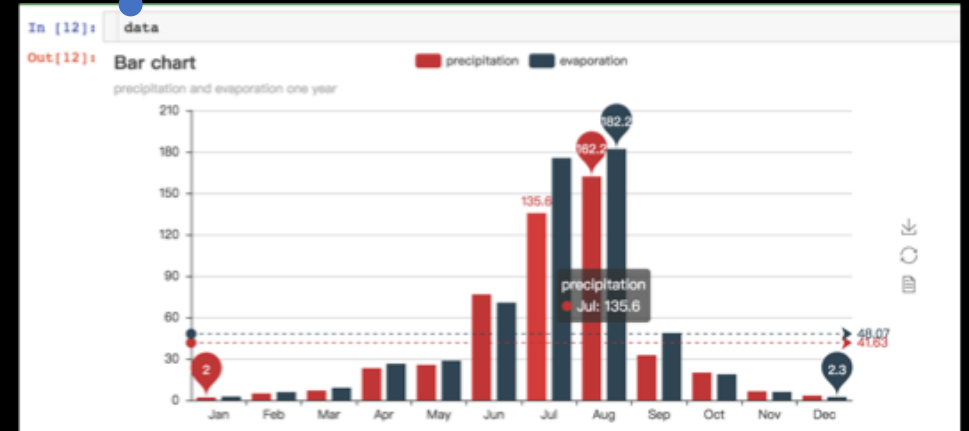
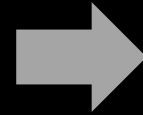
```
<script>
  require.config({});
</script>

<div id="d1"></div>

<script>
require(['echarts'], function(echarts) {
  var chart1 = echarts.init(
    document.getElementById('d1'),
    'light',
    {renderer: 'canvas'});

  var option_1 = {
  };

  chart1.setOption(option_1);
});
</script>
```

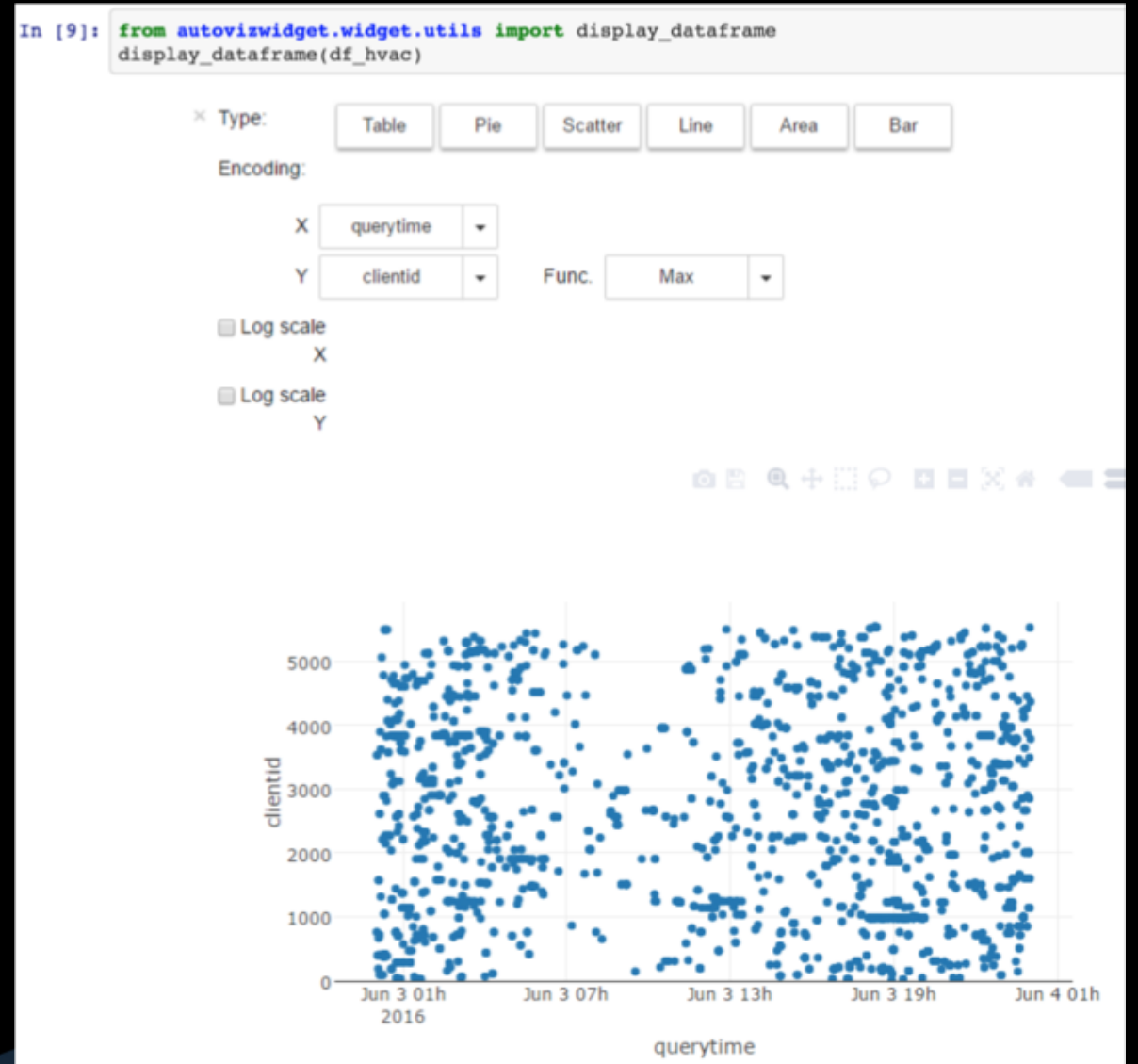




对象自定义可视化

- 基于ipywidgets进一步扩展

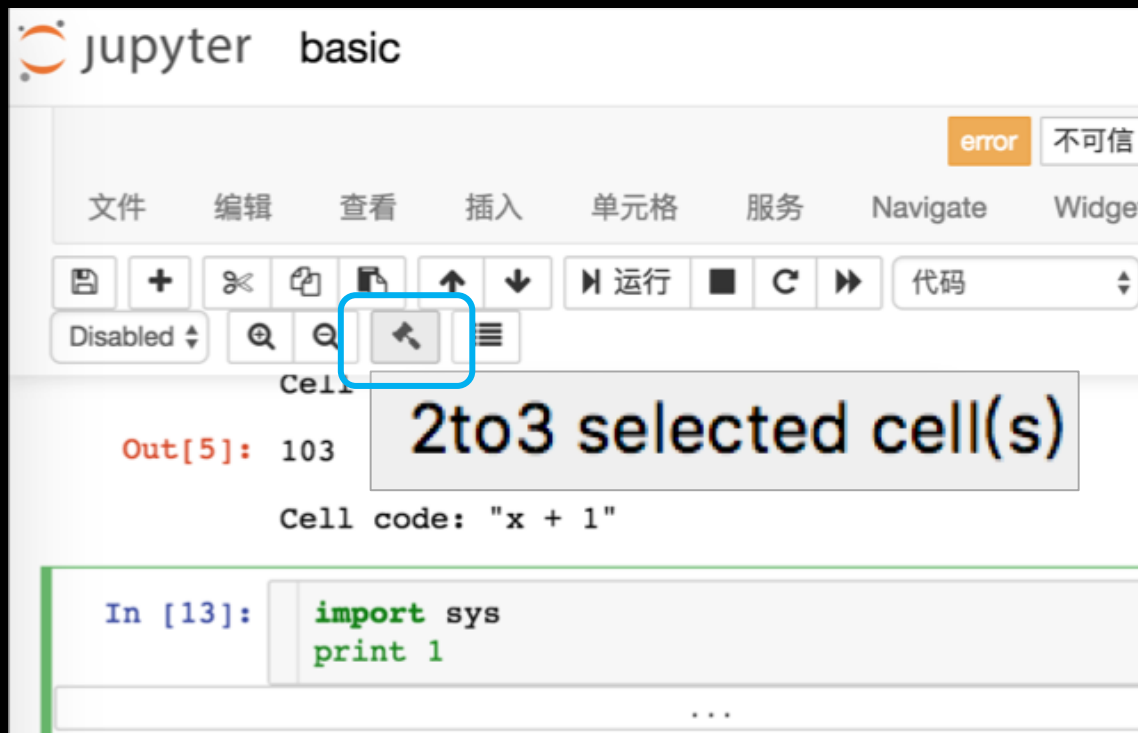
```
from ipywidgets import VBox, Output,  
    Button, HTML, HBox, Dropdown, Checkbox,  
    ToggleButtons, Text, Textarea, Tab  
  
class MyChart(VBox):  
    ....
```





自定义命令

- 如何给工具栏和命令行添加一个功能？





自定义命令

- 如何给工具栏和命令行添加一个功能？

```
%%javascript
var autoformat_cells = function({...});

Jupyter.keyboard_manager.actions.register(
  {
    help: 'convert selected cell(s)',
    icon: 'fa-legal',
    handler: function(evt) { autoformat_cells(); },
  }, 'process_selected_cells', '2to3');
```

```
Jupyter.keyboard_manager.command_shortcuts.add_shortcut('Ctrl+L', function (event)
  autoformat_cells();
  return false;
});
```



构建NB插件

~/.ipython/profile_default/nbconfig/notebook.json

```
{  
  "load_extensions": {  
    "my_ext/main": true  
  }  
}
```

~/Library/Jupyter/nbextensions/myext/main.js

```
define(function(){  
  return {  
    load_ipython_extension: function(){  
      // logic here  
      IPython.keyboard_manager.XXXXX  
    }  
  };  
})
```

ipython-contrib/jupyter_contrib_nbextensions : 一个jupyter NB插件库 (接近60个)

The screenshot shows the Jupyter Nbextensions configuration interface. At the top, there is a 'jupyter' logo, the title 'Nbextensions configuration (more information)', and a '注销' (Logout) button. Below the title is a 'Configurable nbextensions' section with a refresh icon. A checkbox at the top allows disabling configuration for nbextensions without explicit compatibility. A search filter is set to 'by description, section, or tags'. The main area contains a grid of 60 checkboxes, each representing an extension. Some extensions are highlighted in yellow, and 'zenmode' is highlighted in blue. The extensions listed are:

- (some) LaTeX environments for Jupyter
- AutoSaveTime
- Code Font Size
- CodeMirror mode extensions
- datestamper
- Execution Dependencies
- Freeze
- Hide input
- Hinterland
- jupyter-js-widgets/extension
- Live Markdown Preview
- Nbextensions dashboard tab
- Printview
- Runtools
- SKILL Syntax
- spellchecker
- Toggle all line numbers
- 2to3 Converter
- Autoscroll
- Code prettify
- Collapsible Headings
- echarts/main
- Exercise
- Gist-it
- Hide input all
- Initialization cells
- Keyboard shortcut editor
- Load TeX macros
- Nbextensions edit menu item
- Python Markdown
- Scratchpad
- Skip-Traceback
- Split Cells Notebook
- Tree Filter
- AddBefore
- bqplot/extension
- Codefolding
- Comment/Uncomment Hotkey
- Equation Auto Numbering
- Exercise2
- Help panel
- Highlight selected word
- ipyparallel/main
- Launch QTConsole
- Move selected cells
- nbTranslate
- Rubberband
- ScrollDown
- Snippets
- Table of Contents (2)
- Variable Inspector
- Autopep8
- Cell Filter
- Codefolding in Editor
- contrib_nbextensions_help_item
- ExecuteTime
- Export Embedded HTML
- Hide Header
- highlighter
- isort formatter
- Limit Output
- Navigation-Hotkeys
- Notify
- Ruler
- Select CodeMirror Keymap
- Snippets Menu
- table_beautifier
- zenmode



#4. 扩展架构 - NBGrader

- 将Jupyter变成一个考试系统

Manual Grading / ps1 / problem1 / Submission #1

Your function should print [1, 4, 9, 16, 25, 36, 49, 64, 81, 100] for n = 10. Check that it does:

```
In [3]: squares(10)
Out[3]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
In [4]: correct_squares
```

Full credit No credit 0 / 1.0 + 0 (extra credit)

```
"""Check that squares returns the correct output for several inputs"""
assert squares(1) == [1]
assert squares(2) == [1, 4]
assert squares(10) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
assert squares(11) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]
```

```
-----
AssertionError                                Traceback (most recent call last)
<ipython-input-4-c7326d393566> in <module>()
      1 """Check that squares returns the correct output for several inputs"""
----> 2 assert squares(1) == [1]
      3 assert squares(2) == [1, 4]
      4 assert squares(10) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
      5 assert squares(11) == [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121]

AssertionError:
```

jupyter

Files Running Clusters Assignments

Released, downloaded, and submitted assignments.

Released assignments		
ps0	cogsci131	Fetch
ps1	cogsci131	Fetch

Downloaded assignments

There are no downloaded assignments.

Submitted assignments

There are no submitted assignments.



Jupyter生态中的其他开源项目

- 网络与调度：
 - JupyterHub : Jupyter Notebook的网关
 - Jupyter Kernel Gateway : Kernel的网关，转换MQ协议为REST
 - Ipyparallel: 对Ipython Kernel进行集群管理与调度
- 工作流与结果转换：
 - nteract/papermill: 基于NB的工作流参数化运行、分析与结果汇总管理。
 - Nbconvert : NB文件的运行与格式转换工具（如PDF）
- 可视化改进：
 - RISE : 将Jupyter变为PPT播放模式
 - Jupyter Dashboards : 因此代码，调整布局，将NB变成报表样式



3

云服务与Jupyter

使用Jupyter扩展云服务场景、技术方法与策略

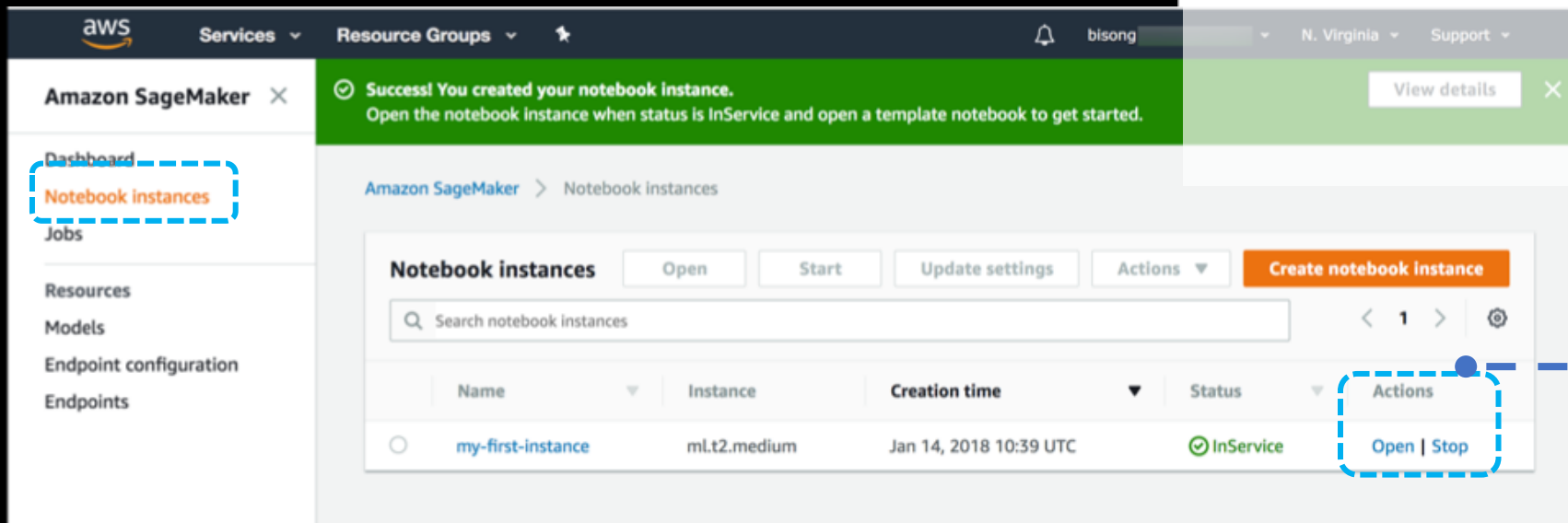


云服务与Jupyter模式

- 托管服务：
 - 提供Notebook的PaaS/SaaS平台，一般支持各种计算运行环境（conda/Spark/TensorFlow等），用户无需关注环境部署等问题，如AWS SageMaker、Google Conlab、Azure Notebook等
- 非托管接口：
 - 支持通过Notebook访问计算平台，但不Host Jupyter环境，如阿里云PyODPS, Azure HDInsight.
- NB增强：
 - 除了环境支持增强外，对NB做了较大交互增强甚至重写，如Google Conlab、PyODPS。
- 阅读分享NB：
 - 强化了展示与分享功能，如mybinder, NBViewer、Google Conlab , Azure Notebook等。

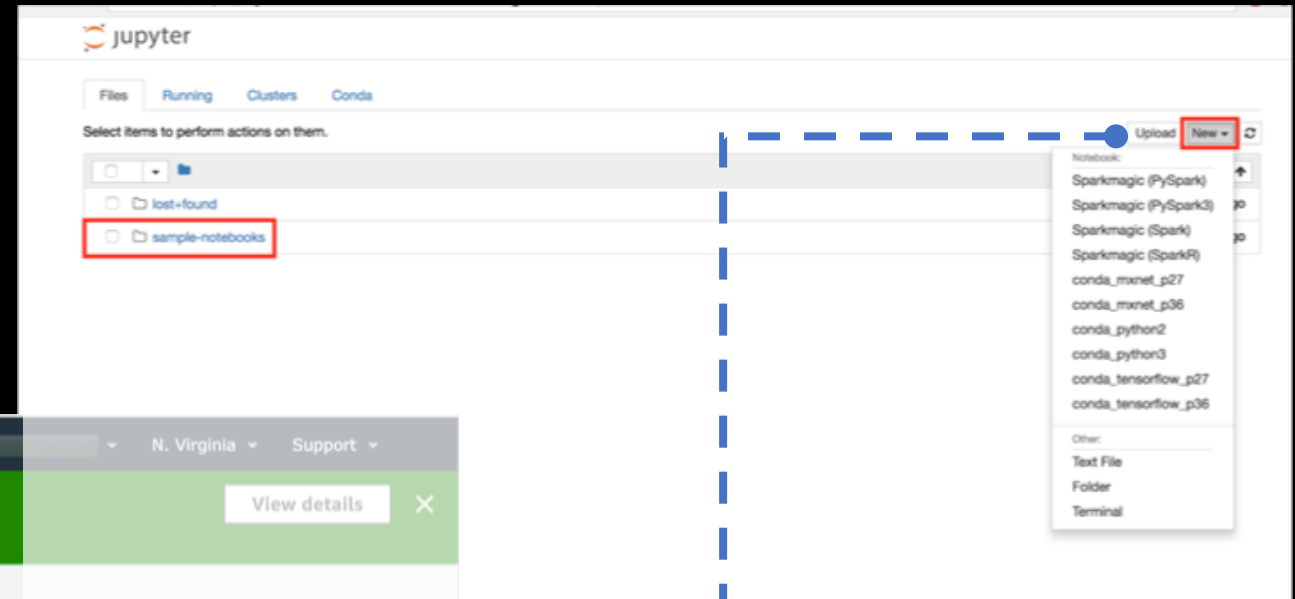
AWS SageMaker

- 全面支持Jupyter的多种科学计算环境
 - PySpark、Spark、SparkR、Tensorflow、MxNet、Conda等
- 支持**基于以上集群的离线任务调度**



The screenshot shows the AWS SageMaker console. A green notification banner at the top states: "Success! You created your notebook instance. Open the notebook instance when status is InService and open a template notebook to get started." Below this, the "Notebook instances" page is visible, featuring a table with the following data:

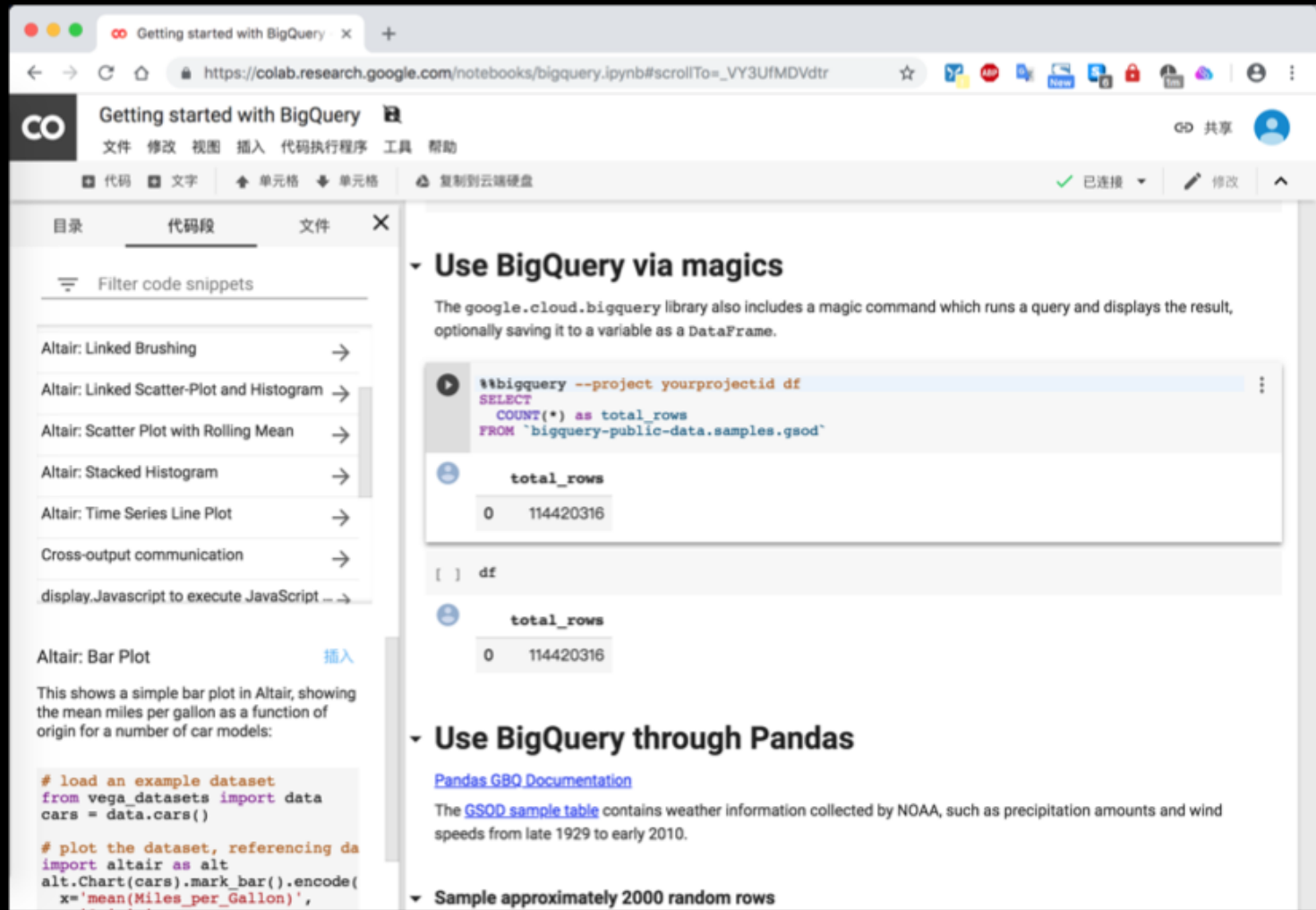
Name	Instance	Creation time	Status	Actions
my-first-instance	ml.t2.medium	Jan 14, 2018 10:39 UTC	InService	Open Stop



The screenshot shows the JupyterLab interface. The "Files" tab is active, displaying a file browser. A folder named "sample-notebooks" is highlighted with a red box. A "New" dropdown menu is open, showing various notebook templates such as "Sparkmagic (PySpark)", "conda_mxnet_p27", and "conda_tensorflow_p36".

Google Colab

- 提供Notebook的科学计算环境
 - 主要TensorFlow、Conda等
- 重写了Jupyter前端
 - **支持多人编辑、表单**
- **运行时间不可靠，但支持本地运行。**
- 存储绑定云盘使用等



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `https://colab.research.google.com/notebooks/bigquery.ipynb#scrollTo=_VY3UfMDVdtr`. The notebook title is "Getting started with BigQuery". The interface includes a menu bar with options like "文件", "修改", "视图", "插入", "代码执行程序", "工具", and "帮助". Below the menu, there are tabs for "代码", "文字", "单元格", and "复制到云端硬盘". The main content area is divided into two sections: "Use BigQuery via magics" and "Use BigQuery through Pandas".

Use BigQuery via magics

The `google.cloud.bigquery` library also includes a magic command which runs a query and displays the result, optionally saving it to a variable as a DataFrame.

```
%%bigquery --project yourprojectid df
SELECT
  COUNT(*) as total_rows
FROM `bigquery-public-data.samples.gsod`
```

total_rows
0
114420316

[] df

total_rows
0
114420316

Use BigQuery through Pandas

[Pandas GBQ Documentation](#)

The [GSOD sample table](#) contains weather information collected by NOAA, such as precipitation amounts and wind speeds from late 1929 to early 2010.

▼ Sample approximately 2000 random rows



Azure Notebook (Preview)



- Jupyter as Service平台
 - 主要Conda、R和F#
 - **免费4GB内存+1GB存储**
- 内置JupyterLab、RISE等

The screenshot shows a web browser window displaying the Microsoft Azure Notebooks interface. The browser address bar shows the URL `https://notebooks.azure.com/Microsoft/libraries/samples/html/Cr...`. The page title is "Microsoft Azure Notebooks Preview" with a "Sign In" button. Below the navigation bar, there are links for "Libraries", "What's New", "Status", and "Help".

The main content area displays a Jupyter notebook cell with the following code:

```
In [5]: ggplot(aes(x='time', y='temp', color='activ'), data=df) + geom_point()
```

The output of the code is a scatter plot showing the relationship between "time" (x-axis, ranging from -500 to 2500) and "temp" (y-axis, ranging from 36.0 to 38.5). The data points are colored based on the "activ" variable, with a legend on the right indicating "activ" values of 0.0 (black) and 1.0 (blue). The plot shows a clear upward trend in temperature over time, with a significant increase in temperature starting around time 1000. The blue points (activ=1.0) are generally higher on the y-axis than the black points (activ=0.0).

Below the plot, the output is displayed as:

```
Out[5]: <ggplot: (8791972548737)>
```

Textual analysis of the plot is provided below the output:

It also appears that all else being equal, Beaver 2 is warmer than Beaver 1.

Notice that some trends seem to apply to a subset of the data. For example, if the measurement is from Beaver 2 and the beaver was active, then the body temperature is likely to be above 37.5:



Alicloud PyODPS/Log Service



- PyODPS: 类似PySpark/SparkMagic 提供DataFrame子集接口。
- 对Jupyter做了较多交互性增强。
- 不提供Jupyter的托管服务。

The screenshot shows a Jupyter notebook interface with a modal dialog box titled "Progress of 'DataFrame Operation[sum]'" overlaid on top. The dialog indicates that the operation is running and shows a progress bar for "Instance 0: Running" which is at 100.00%. Below the progress bar, it lists "AnonymousSQLTask: RUNNING" and two sub-tasks: "M1_Stg1_job0" and "R2_1_Stg1_job0", both with 100% completion. A "Logview" link is present at the bottom of the dialog. The background shows the Jupyter notebook with code cells and a partially visible progress bar for the main operation.

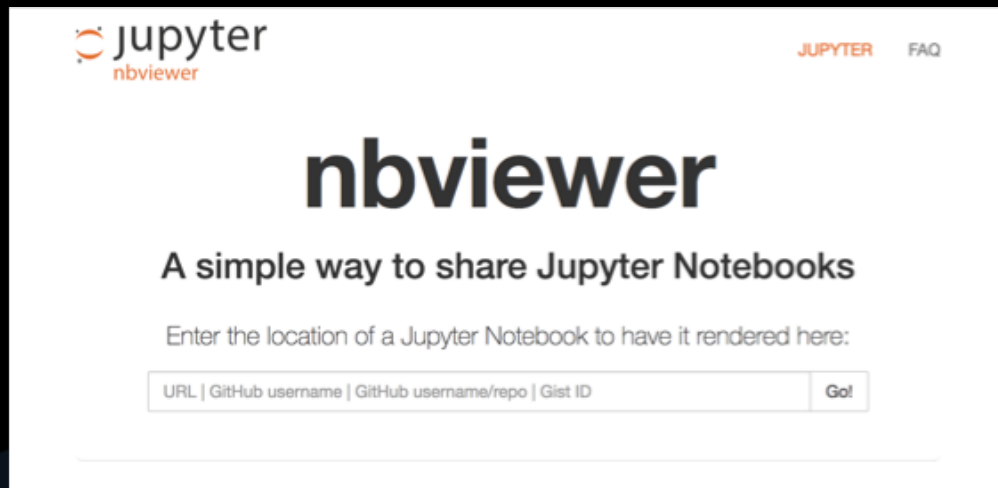
The screenshot shows a Jupyter notebook interface with a configuration dialog box titled "jupyter log service (未保存改变)". The dialog has tabs for "账户" and "高级". Under "账户", there are fields for "默认区域" (Default Region) set to "上海", "密钥ID" (Access Key ID) set to "ABCD12314", and "密钥Key" (Access Key Secret) set to "QWERTASAF". A "确定" (Confirm) button is at the bottom. Below the dialog, the notebook shows a code cell with a SQL query: `population > 1000 | SELECT province AS "省份", COUNT(1) AS "订单" GROUP BY province DESC`. The output shows a table with columns "省份" and "订单":

省份	订单
0 上海	10000
1 江苏	2000
2 山东	4900

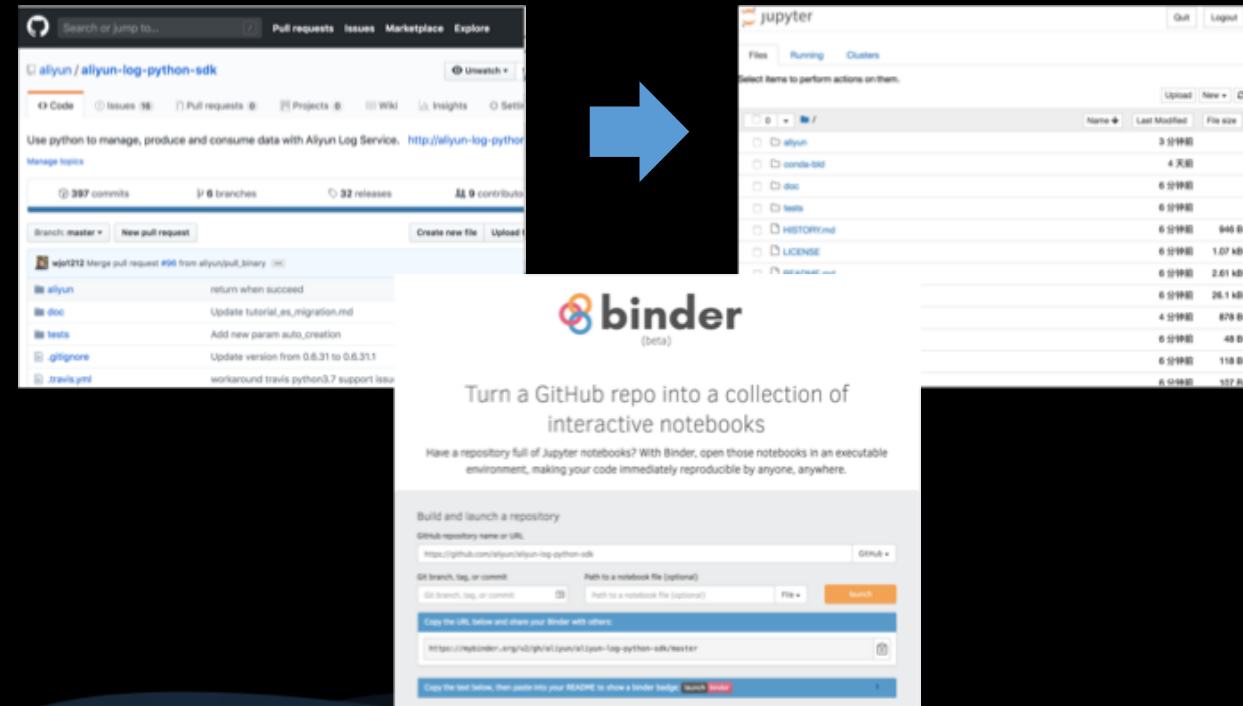
MyBinder/NBViewer

MyBinder：提供将Github项目转化为一个Docker环境下可运行的Jupyter项目

NBViewer：提供URL到Notebook展示的服务。



The image shows the NBViewer web interface. At the top left is the 'jupyter nbviewer' logo. On the right, there are links for 'JUPYTER' and 'FAQ'. The main heading is 'nbviewer' in a large, bold font. Below it is the subtitle 'A simple way to share Jupyter Notebooks'. A text prompt says 'Enter the location of a Jupyter Notebook to have it rendered here:'. Below this is a text input field with the placeholder text 'URL | GitHub username | GitHub username/repo | Gist ID' and a 'Go!' button to the right.



The diagram illustrates the MyBinder workflow. It starts with a screenshot of a GitHub repository page for 'aliyun / aliyun-log-python-sdk'. A blue arrow points from this repository to a screenshot of the Binder website. The Binder website has the heading 'Turn a GitHub repo into a collection of interactive notebooks' and a sub-heading 'Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.' Below this is a form to 'Build and launch a repository' with fields for 'GitHub repository name or URL' (containing 'https://github.com/aliyun/aliyun-log-python-sdk'), 'GitHub branch, tag, or commit', and 'Path to a notebook file (optional)'. A 'Search' button is next to the last field. Below the form, a generated URL is shown: 'https://mybinder.org/v2/gh/aliyun/aliyun-log-python-sdk/master'. A second blue arrow points from the Binder website to a screenshot of a JupyterLab interface. The JupyterLab interface shows a file browser with a tree view containing folders like 'aliyun', 'conda-reqs', 'doc', 'tests', and files like 'HISTORY.md' and 'LICENSE'. A table on the right lists these files with their last modified dates and sizes.



4

Demo

演示扩展Jupyter、赋能云服务的一些场景





THANK YOU



本人微信



Github 下载PPT



日志服务钉钉群

