

大数据方向项目报告

(简略版)

北京邮电大学 凌杰

2019年9月

目录

1. 项目背景.....	3
1.1 大数据定义.....	3
1.2 大数据发展史.....	3
2. 项目回顾.....	4
2.1 Git 与 GitHub.....	4
2.2 Docker 与容器化技术.....	5
2.3 Flask 与 RESTful API.....	6
2.4 Mnist 手写数字识别.....	7
2.5 NoSQL 非关系型数据库.....	7
3. 项目收获与感想.....	8

1. 项目背景

1.1 大数据定义

随着高速发展的信息技术，不断扩张的数据库容量，互联网作为信息传播和再生的平台，“信息泛滥”、“数据爆炸”等现象不绝于耳，海量的数据信息使得人们难以做出快速的抉择。全球互联网发展速度达到每半年就增加一倍，以《纽约时报》为例，在 20 世纪 60 年代的内容版面不过十几二十页左右，而如今已达到一百到二百页。于此情境下，大数据技术及其衍生出的一系列技术概念渐渐成了产业界关注的焦点。

究竟何谓大数据？在 2001 年左右，Gartner 就大数据提出了如下定义（目前仍是关于大数据的权威解释）：大数据指高速 (Velocity) 涌现的大量 (Volume) 的多样化 (Variety) 数据。这一定义表明大数据具有 3V 特性：

大数据的“大”首先体现在数据量上。在大数据领域，我们需要处理海量的低密度的非结构化数据，数据价值可能未知，例如 Twitter 数据流、网页或移动应用点击流，以及设备传感器所捕获的数据等等。在实际应用中，大数据的数据量通常高达数十 TB，甚至数百 PB。

大数据的“高速”指高速接收乃至处理数据—数据通常直接流入内存而非写入磁盘。在实际应用中，某些联网的智能产品需要实时或近乎实时地运行，要求基于数据实时评估和操作，而大数据只有具备“高速”特性才能满足这些要求。

多样化是指可用的数据类型众多。通常来说，传统数据属于结构化数据，能够整齐地纳入关系数据库中。随着大数据的兴起，各种新的非结构化数据类型不断涌现，例如文本、音频和视频等等，它们需要经过额外的预处理操作才能真正提供洞察和支持性元数据。

简而言之，大数据指越来越庞大、越来越复杂的数据集，特别是来自全新数据源的数据集，其规模之大令传统数据处理软件束手无策，却能帮助我们解决以往非常棘手的业务难题。

1.2 大数据发展史

2005 年左右，人们开始意识到用户在使用 Facebook、YouTube 以及其他在线服务时生成了海量数据。同一年，专为存储和分析大型数据集而开发的开源框架 Hadoop 问世，NoSQL 也在同一时期开始慢慢普及开来。

Hadoop 及后来 Spark 等开源框架的问世对于大数据的发展具有重要意义，正是它们降低了数据存储成本，让大数据更易于使用。在随后几年里，大数据数量进一步呈爆炸式增长。时至今日，全世界的“用户”—不仅有人，还有机器—仍在持续生成海量数据。

2. 项目回顾

2.1 Git 与 GitHub

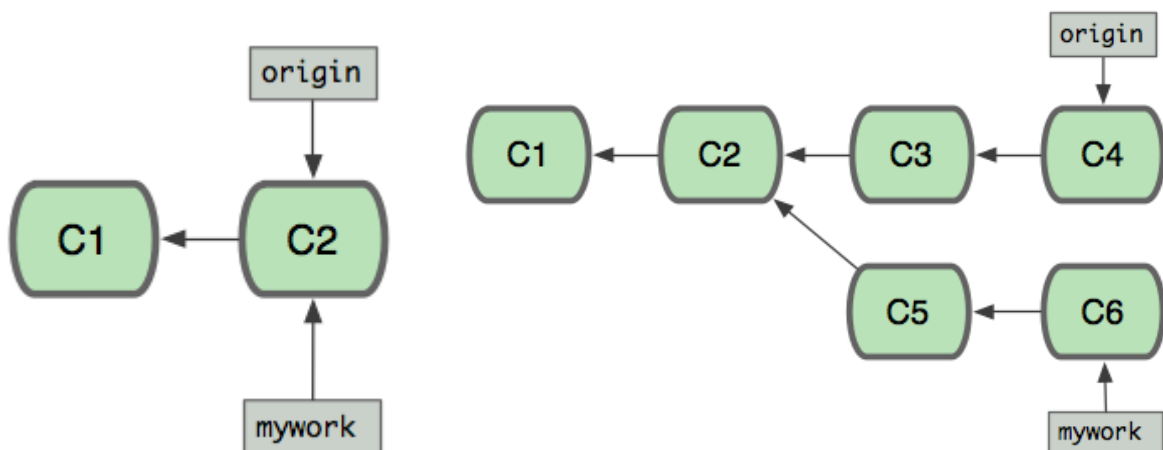
本项目中，我主要使用 Git 作为代码版本管理工具。它能自动帮我记录每次文件的改动，还可以让他人协作编辑，甚至编辑完也不需要把文件传来传去，用一条指令将各自的修改合并即可。如果想查看某次改动，只需要在软件里瞄一眼就可以。

```
Jerison@LAPTOP-K3CULHRE MINGW64 ~/Git_Sample (master)
$ git remote add origin git@github.com:Jerison/Git-Test.git

Jerison@LAPTOP-K3CULHRE MINGW64 ~/Git_Sample (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 471 bytes | 58.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To github.com:Jerison/Git-Test.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

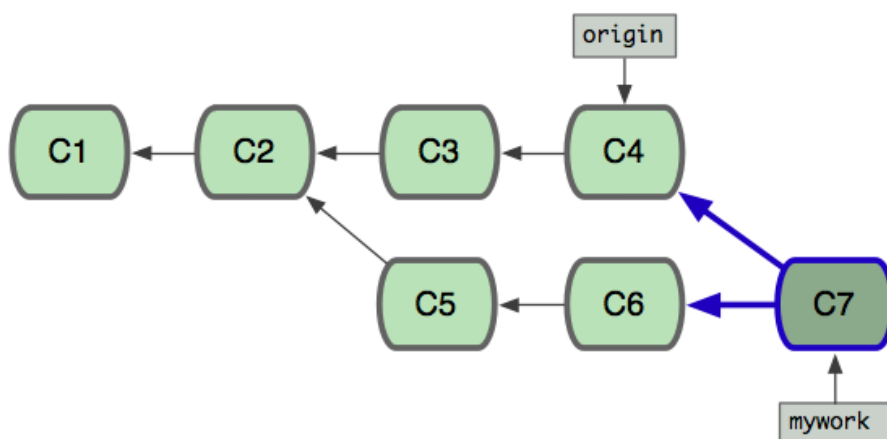
(将本地库 push 到远程的操作)

同时，在项目过程中，我对之前不太用的 rebase 操作有了更深的理解。



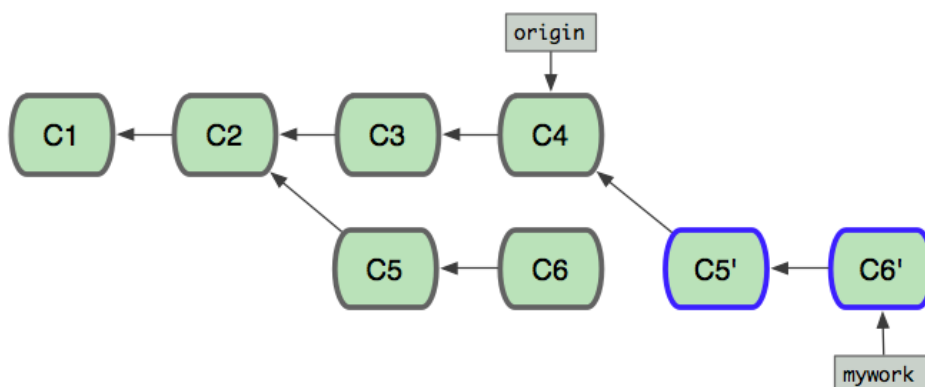
如上图所示，当 mywork 分支和 origin 分支各自被不同人修改，C2 此时分叉成了 C4 和 C6。在这里，原本我（mywork）可能会用“pull”命令把“origin”分支上的修改拉下来并且和我的修改合并；结果看起来就像一个新的“合并的提交”(merge commit)，如下图：

git merge



而现在，经过项目的学习，我学会了使用 git rebase 使得分支历史看起来像没有经过任何合并一样。如下图所示：

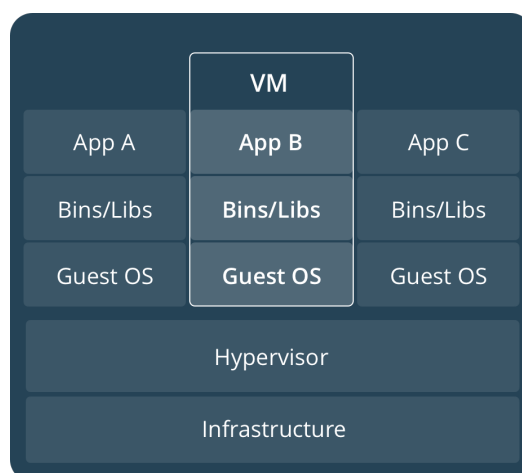
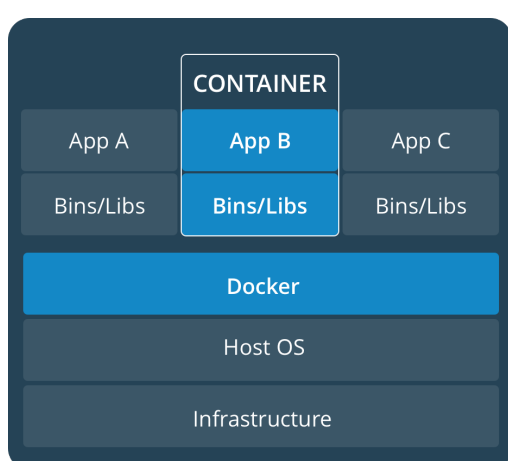
git rebase

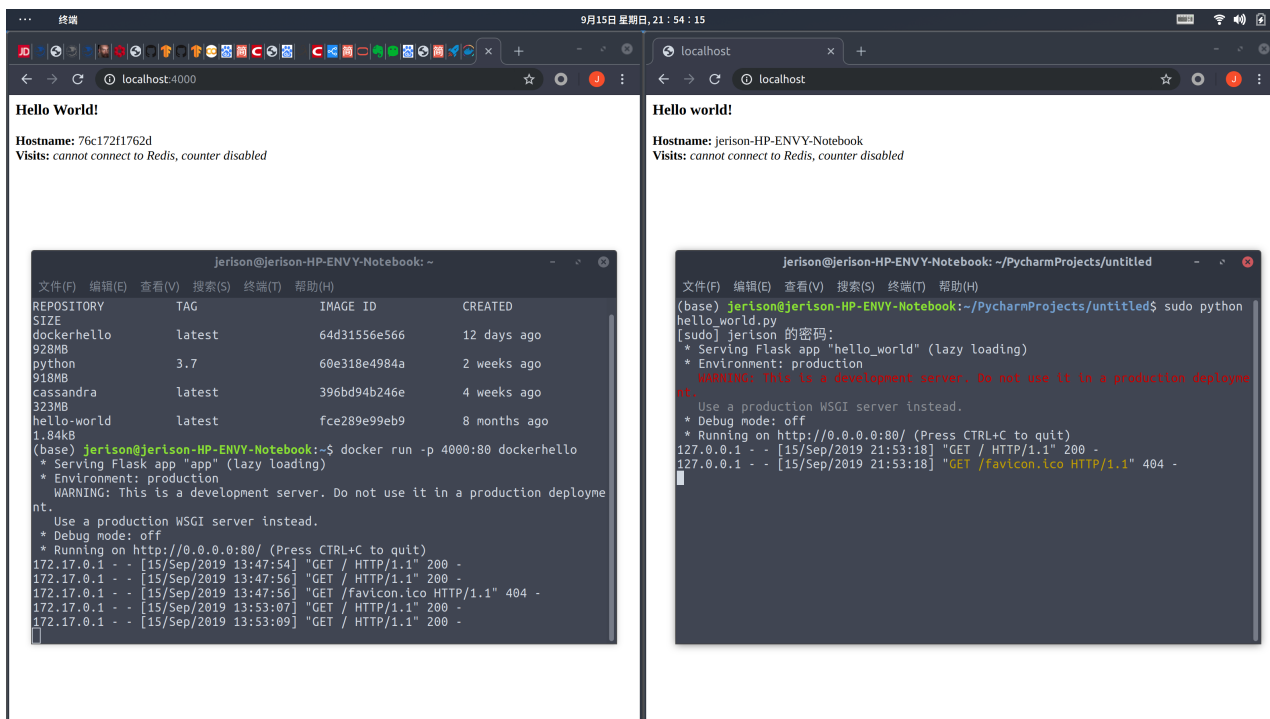


以我个人的理解，这相当于是把 origin 分支上的最新修改拉取下来作为 base（原先是 C2），于此基础上进行 C5、C6 的修改操作。这对多人项目协作而言是十分实用的利器。

2.2 Docker 与容器化技术

在本项目中，我第一次接触到了与 Docker 有关的容器化技术，了解到它是有别于 VMware 等传统虚拟机的轻量级虚拟化方案。（如下图）





于此基础上搭建了属于自己的第一个微服务。

如上图，从容器内和容器外去访问同一个服务，其 hostname 是不同的。这体现了容器技术虚拟化的特点。

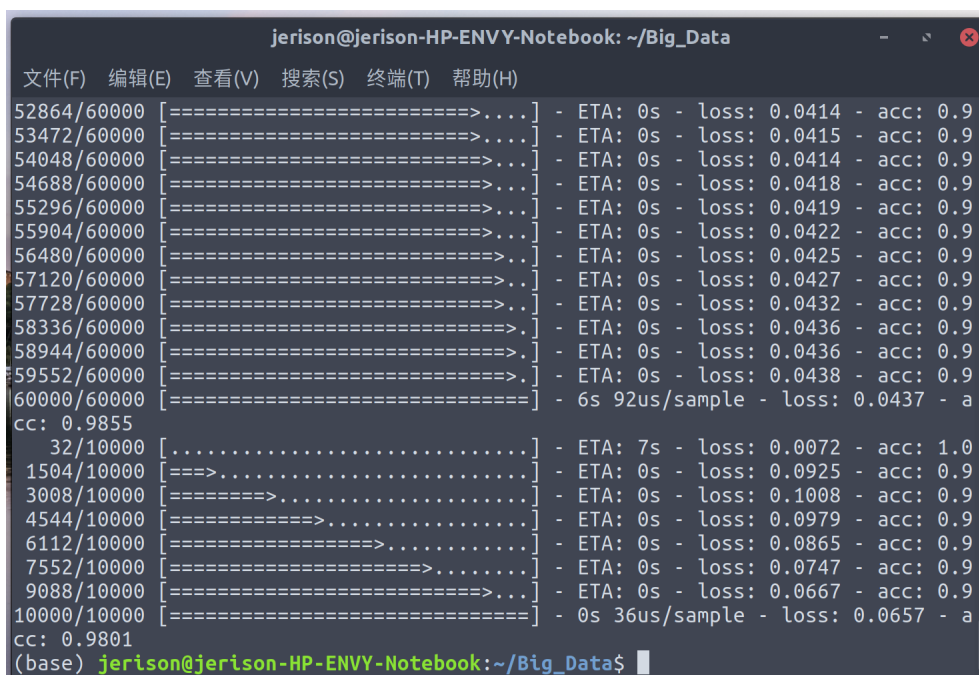
2.3 Flask 与 RESTful API

近年来移动互联网的发展，前端设备层出不穷（手机、平板、桌面电脑、其他专用设备.....），因此，必须有一种统一的机制，方便不同的前端设备与后端进行通信,于是 RESTful 诞生了，它可以通过一套统一的接口为 Web，iOS 和 Android 提供服务。REST，即 Representational State Transfer 的缩写。直接翻译的意思是"表现层状态转化"。它是一种互联网应用程序的 API 设计理念：URL 定位资源，用 HTTP 动词（GET,POST,DELETE,DETC）描述操作。本项目中，我学会了搭建与调用属于自己的 RESTful API，并掌握了 GET、POST、DELETE 等主要方法。（如下图为 GET 方法）

```
hello_world.py
1  #!/usr/bin/python
2  from flask import Flask, jsonify
3
4  app = Flask(__name__)
5
6  tasks = [
7      {
8          'id': 1,
9          'title': 'u'Buy groceries',
10         'description': 'u'Milk, Cheese, Pizza, Fruit, Tylenol',
11         'done': False
12     },
13     {
14         'id': 2,
15         'title': 'u'Learn Python',
16         'description': 'u'Need to find a good Python tutorial on the web',
17         'done': False
18     }
19 ]
20
21 @app.route('/todo/api/v1.0/tasks', methods=['GET'])
22 def get_tasks():
23     return jsonify({'tasks': tasks})
24
25 if __name__ == '__main__':
26     app.run(debug=True)
```

2.4 Mnist 手写数字识别

Mnist 可以说是机器学习界的“Hello_Word”。在本项目中，我主要需要将 Mnist 应用部署到容器里面，用户通过 curl -XPOST 命令提交带有手写体的数字图片。如下图，训练后的模型识别准确度可以达到 98%。



```
jerison@jerison-HP-ENVY-Notebook: ~/Big_Data
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
52864/60000 [=====>...] - ETA: 0s - loss: 0.0414 - acc: 0.9
53472/60000 [=====>...] - ETA: 0s - loss: 0.0415 - acc: 0.9
54048/60000 [=====>...] - ETA: 0s - loss: 0.0414 - acc: 0.9
54688/60000 [=====>...] - ETA: 0s - loss: 0.0418 - acc: 0.9
55296/60000 [=====>...] - ETA: 0s - loss: 0.0419 - acc: 0.9
55904/60000 [=====>...] - ETA: 0s - loss: 0.0422 - acc: 0.9
56480/60000 [=====>...] - ETA: 0s - loss: 0.0425 - acc: 0.9
57120/60000 [=====>...] - ETA: 0s - loss: 0.0427 - acc: 0.9
57728/60000 [=====>...] - ETA: 0s - loss: 0.0432 - acc: 0.9
58336/60000 [=====>...] - ETA: 0s - loss: 0.0436 - acc: 0.9
58944/60000 [=====>...] - ETA: 0s - loss: 0.0436 - acc: 0.9
59552/60000 [=====>...] - ETA: 0s - loss: 0.0438 - acc: 0.9
60000/60000 [=====] - 6s 92us/sample - loss: 0.0437 - a
cc: 0.9855
 32/10000 [.....] - ETA: 7s - loss: 0.0072 - acc: 1.0
1504/10000 [====>.....] - ETA: 0s - loss: 0.0925 - acc: 0.9
3008/10000 [=====>.....] - ETA: 0s - loss: 0.1008 - acc: 0.9
4544/10000 [=====>.....] - ETA: 0s - loss: 0.0979 - acc: 0.9
6112/10000 [=====>.....] - ETA: 0s - loss: 0.0865 - acc: 0.9
7552/10000 [=====>.....] - ETA: 0s - loss: 0.0747 - acc: 0.9
9088/10000 [=====>.....] - ETA: 0s - loss: 0.0667 - acc: 0.9
10000/10000 [=====] - 0s 36us/sample - loss: 0.0657 - a
cc: 0.9801
(base) jerison@jerison-HP-ENVY-Notebook: ~/Big_Data$
```

2.5 NoSQL 非关系型数据库

随着大数据时代的到来，越来越多的网站、应用系统需要支撑海量数据存储，高并发请求、高可用、高可扩展性等特性要求，传统的关系型数据库在应付这些调整已经显得力不从心，暴露了许多能以克服的问题。由此，各种各样的 NoSQL（Not Only SQL）数据库作为传统关系型数据的一个有力补充得到迅猛发展。

本项目中，在将 Mnist 部署在 Docker 中后，我需要考虑的是如何将 MNIST 中用户每次提交的图片到文件名、识别的文字和时间戳信息，记录到一个非关系型数据库中。

3. 项目收获与感想

在加入本项目之前，大数据于我而言只是个略空泛而模糊的概念。在经过这一个月对产业界各类大数据技术的学习后，我方才领略到大数据技术领域的整个 Landscape，不得不为其中绝妙的技术细节、极其丰富的生态与广阔的应用前景所深深折服。

前期我的主要学习对象是 Docker 及其相关技术。近年来互联网技术行业普遍追随“轻量化”的概念，但了解到 Docker 的单个容器“系统”能小至 KB 级别时，我还是大吃了一惊，瞬间便对这样一款跨平台的容器引擎产生了兴趣。如今在我眼里，Docker 是和 Virtualenv 一样能大大提高开发体验和开发效率的神器。

而后学习的 RESTful API 和 NoSql 等技术则像是一盏盏明灯，为我的技术学习道路与职业发展道路指明了一条方向。平常在学校跟随课本学习特别容易陷入到一种困惑中去：“这些技术怎么用？有什么用？产业界如今是怎么使用这些技术的？”常年的“闭门造车”让我很少能看到校园外货真价实的技术实战场景，而本项目便是替我在象牙塔封闭的墙面上凿开了一扇窗。

因而最重要的，是本次项目让我窥探到了当今产业界的繁荣样貌，从而彻底激发了我对大数据技术领域的兴趣。如果说之前只是被热度裹挟着去了解这一领域，那现在我愿意一心一意继续深挖其中的更多更深的技术。因为这不仅有用，更有趣，钻入其中更能收获一种前所未有的好奇心被满足的痛快感与成就感。

最后，本次项目亦是培养了我一项十分关键的能力——独立开发能力。我们常常打趣说当今软件工程师是在“面向 Google 编程”，虽是句玩笑话却体现出自学能力和独立解决问题的能力对一个技术从业者而言是多么重要。我很开心我的这一能力能在此过程中得到培养。也更有信心和兴趣投入到之后的技术学习中去。