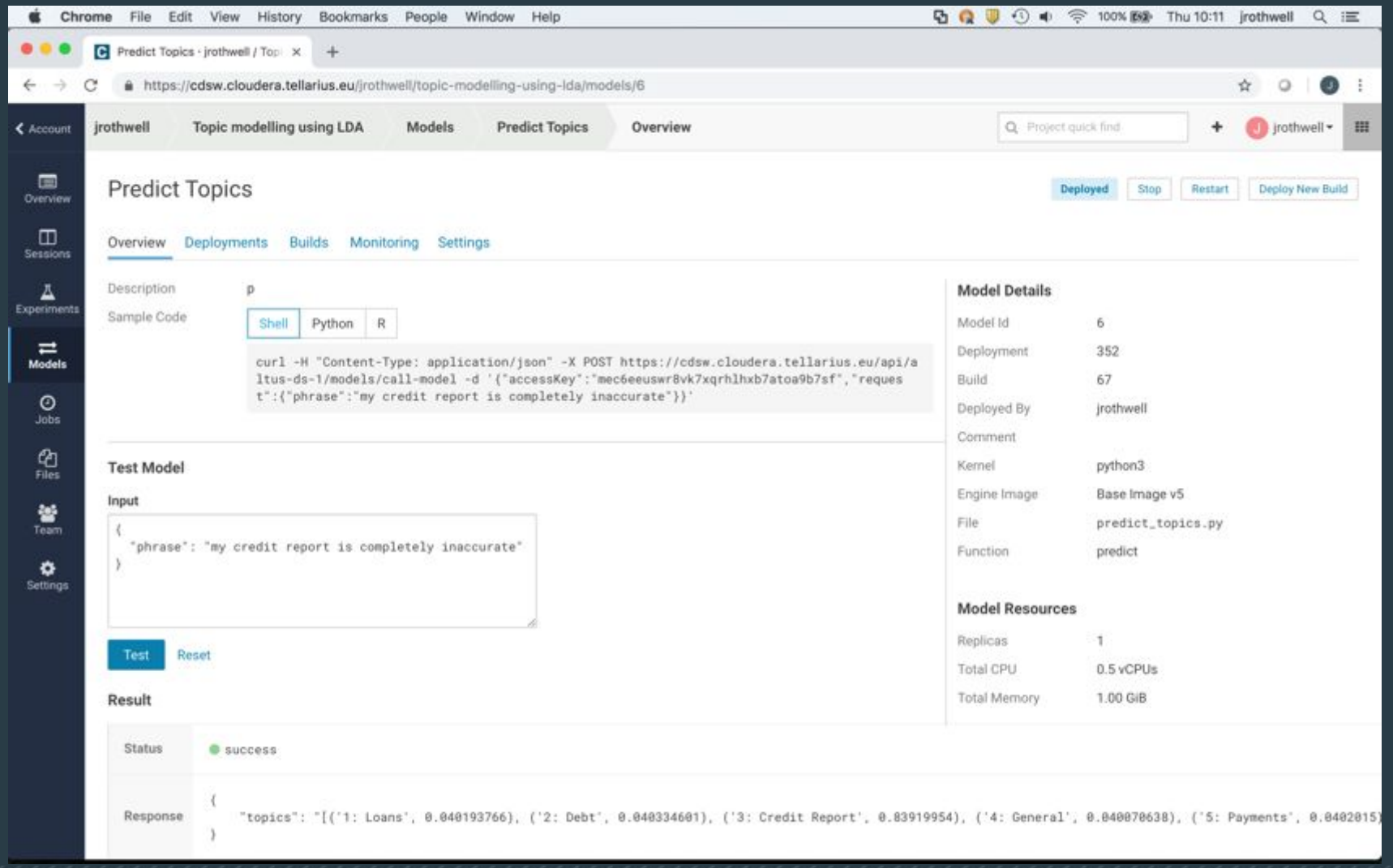
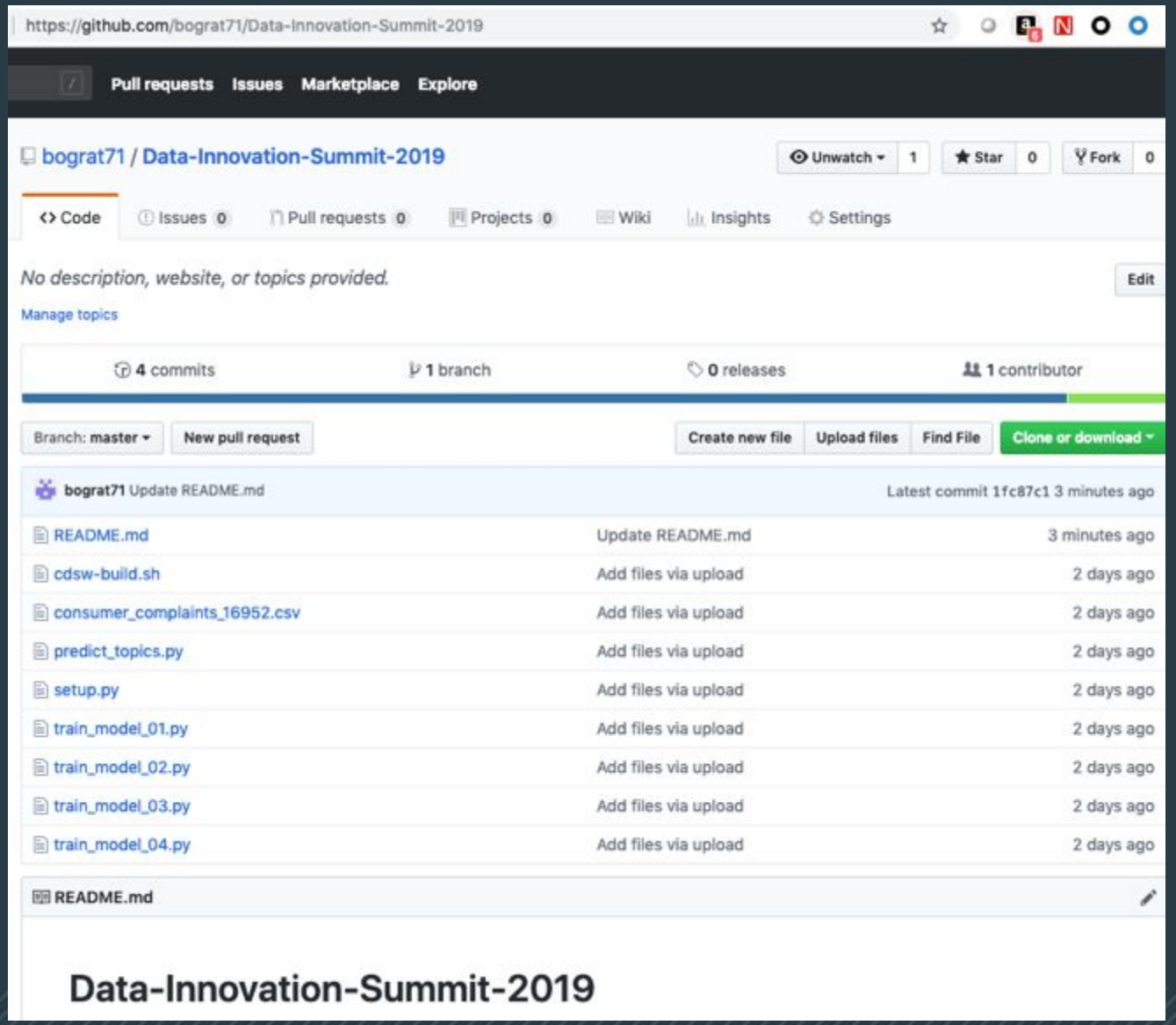
DEMO BUILD

The Challenge



Project Flow

- Acquire Data
- Create new Python project and Upload data file
- Step 01 - populate source dataframe and series
- Step 02 - Pre-process the data: lowercase, tokenise, stop-words, stemming
- Step 03 - Build the LDA model
- Step 04 - Visualise the model
- Step 05 - Write the prediction function
- Step 06 - Deploy the model



Logistics

- Have you brought your laptop and want to do the hands on lab?
- Collect a UserID and Password from one of the team:

```
Trainingxx/   bwh3PcZX6ij6puz
```

- Data Science Workbench URL: https://cdsw.cloudera.tellarius.eu
- Github repo: https://github.com/bograt71/Data-Innovation-Summit-2019

  https://bit.ly/2UBfETt

Acquire Data

- https://catalog.data.gov/dataset/consumer-complaint-database
- >600MB
- We will run with a subset for today : consumer_complaints_16952.csv
- Open the csv file and take a quick look
  - It is already labelled – we will be ignoring that for the demo
  - We are just going to use the Customer complaint narrative field
  - Notice there is a fair amount of superfluous text in there

Create a new Blank Project and open a Workbench

- Open a Python 3 Workbench
- Upload the source data file
- Create a Readme.md
- Create a setup.py to install the required packages
- setup.py is here: setup.py
- Create a “models” folder where we will store the trained model

Step 01 – populate source dataframe and series

- Create a file called train_model.py
  - this will hold all our code for data prep, model training and visualisation
  - We will be incrementally adding code fragments to this file
- Import packages
- Read complaints csv into a dataframe and extract the complaints narrative into a series
- Code is here: train_model_01.py
- Take a look at the dataframe and the series

```
df.head(5), doc_set, doc_set[0]
```

Step 02 - Pre-process data

- Build a list of tokenised (and “cleansed”) documents
  - Loop through each row in the doc_set
    - Lower case it
    - Tokenise it
    - Remove stop words (I, on, my, and …)
    - Stem to the word root (change,changing,changed -> chang)
- Code is here: train_model_02.py
- Now compare your source complaint to the pre-processed one

  `doc_set[0], texts[0]` (stopped words and stemmed)

Step 03 - Build the LDA model

- turn our tokenized documents into a id <-> term dictionary
- convert tokenized documents into a document-term matrix
- generate LDA model
- save the trained model
- Take a look at the term dictionary and document-term matrix
  - dictionary[0], len(corpus), corpus[0]
- Code is here: train_model_03.py

Step 04 - Visualise the model

- Look at the term composition of the main topics in the model
- Use pyLDAvis to visualise the model to aid labelling of the topics
- Code is here: train_model_04.py

Step 05 - Predict topics

- Create the predict_topics.py to receive a phrase and return the topic list
- Load saved model
- Pre-process the phrase
  - Lowercase, tokenise, stop_words, stemming
- Assemble a bag of words (bow) from the pre-processed phrase
- Score the bow with the LDA model
- Return list of topics
- Try the sample phrases
- Code is here: predict_topics.py

Step 06 - Deploy the predict_topics.py

- First create the cdsw-build.sh script
  - Include the contents of setup.py (i.e. import all necessary packages)
- Open a terminal to your K8S container and change permissions:

```
chmod u+x /home/cdsw/cdsw-build.sh
```

- Code fragment here: cdsw-build.sh

Step 06 - Deploy the predict_topics.py

- Name: Predict topics
- Description: Topic modelling with LDA
- File: predict_topics.py
- Function: predict
- Example Input: { "phrase": "my credit report is completely inaccurate"}
- Kernel: Python 3
- Engine Profile: 0.5 vCPU/1 GiB memory

WELL DONE

cloudera