Scalable Computing for Individuals

Joshua Cook

November 2, 2016

Scalable Computing for Individuals

Problem: Medium-Sized Data

- Kaggle Problems; Datasets from UCI
- Small enough to work with using standard database tools (Postgres, Mongo)
- Large enough to be unwieldy; feature engineering and training is extremely slow
- Advantage of working as an individual can be lost (creativity, rapid innovation)
- Especially, difficulties in using Jupyter with medium to large data sets

Solution: Infrastructure as Code

Use `docker` and `docker-compose` to define a multi-container system for processing data.
Considering Docker best-practice, one process per container, our system uses the following container types:

Jupyter	primary interface to system
Postgres	database
Redis	memory cache
Webserver	basic webserver designed for monitoring worker health
Worker	dedicated python processor

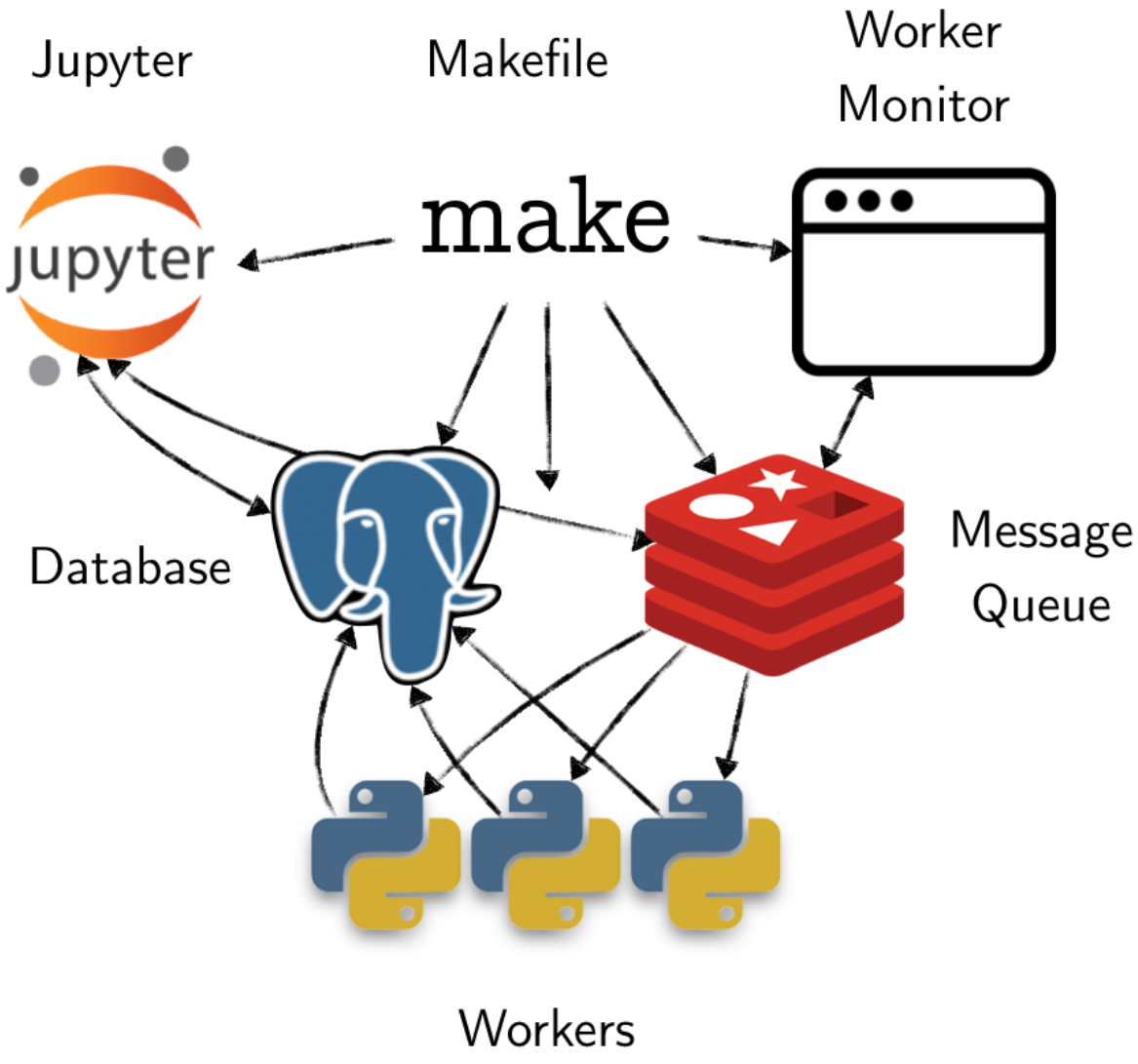

Figure 1: Infrastructure

docker-compose.yml

```
jupyter:
  build: docker/jupyter
  restart: always
  links:
    - redis
    - postgres
  volumes:
    - .:/home/jovyan/work
  ports:
    - 8003:8888

postgres:
  build: docker/postgres
  volumes:
    - .:/home
  volumes_from:
    - postgresdata

postgresdata:
  image: postgres
  command: echo 'Data Container for PostgresDB'
  volumes:
    - /var/lib/postgresql
    - /data/postgres

redis:
  image: redis
  volumes_from:
    - redisdata

redis:
  image: redis
  volumes_from:
    - redisdata

redisdata:
  image: redis
  command: echo 'Data Container for Redis'
  volumes:
    - /data/redis

rq:
  build: docker/python
  links:
    - redis
    - postgres
  restart: always
  entrypoint: ["rq", "worker", "-c", "lib.conf.rq_settings"]
  volumes:
    - .:/usr/src/app

webserver:
  build: docker/python
  restart: always
  links:
    - redis
    - postgres
  volumes:
    - .:/usr/src/app
  ports:
    - 8002:8000
  entrypoint: ["python", "-m", "main"]
```

Controlling the System

Launching the System

Launch triggers in postgres image

Queueing Tasks