# Lecture 15: Cloud Computing II (Prediction)

# Note

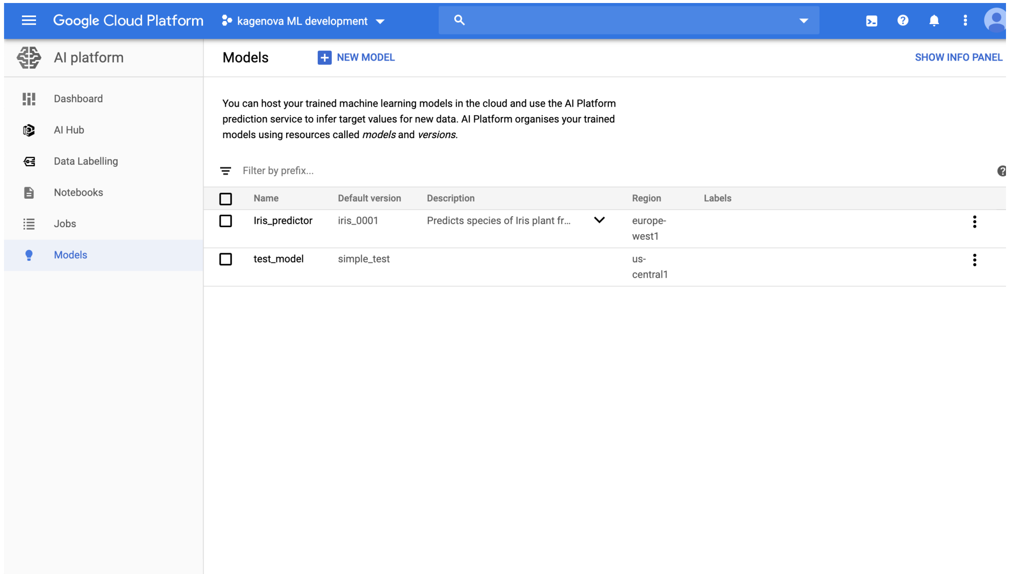
You can do all I am doing from the command line using the `gcloud` and `gsutil` Command Line Interfaces (CLIs).

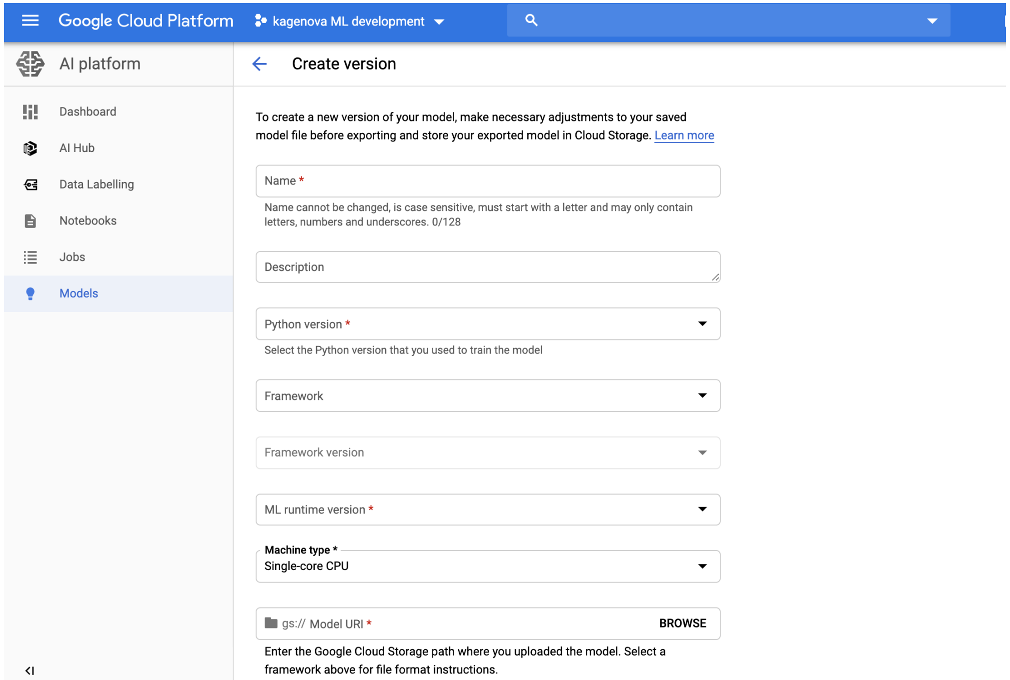
I'm using the web UI here to make it clearer.

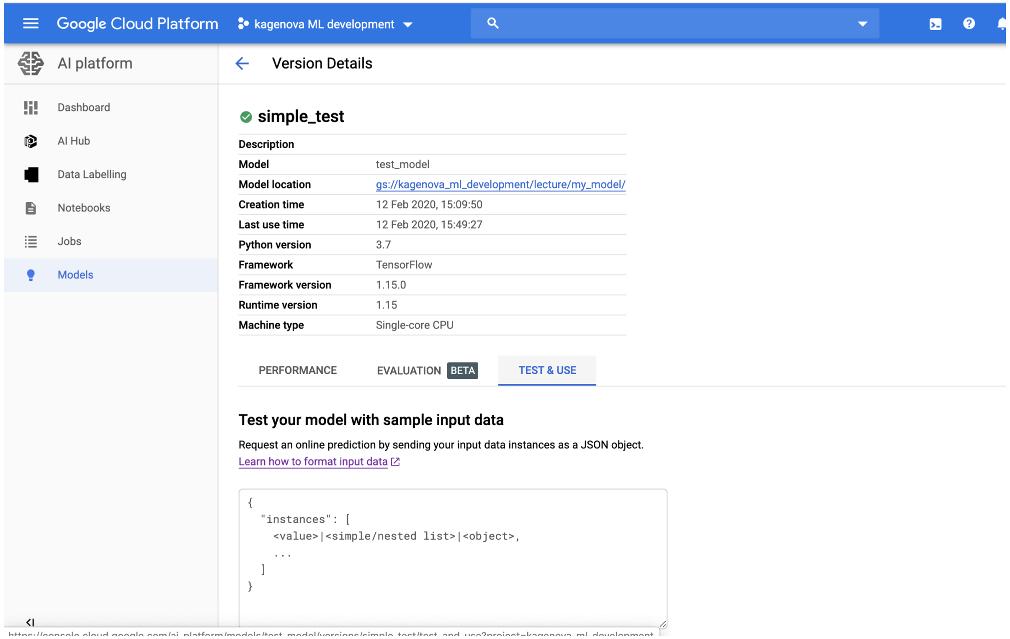
I need to let other people use this model → AI platform

# I need to let other people use this model → AI platform

- Can use TensorFlow serving
- Much better to use AI platform as the serving will scale and control access


Google Cloud Platform
kagenova ML development
AI platform
Models
NEW MODEL
SHOW INFO PANEL
Dashboard
AI Hub
Data Labelling
Notebooks
Jobs
Models
You can host your trained machine learning models in the cloud and use the AI Platform prediction service to infer target values for new data. AI Platform organises your trained models using resources called models and versions.
Filter by prefix...
Name
Default version
Description
Region
Labels
Iris_predictor
iris_0001
Predicts species of Iris plant fr...
europe-west1
test_model
simple_test
us-central1


Google Cloud Platform
kagenova ML development
AI platform
Create version
Dashboard
AI Hub
Data Labelling
Notebooks
Jobs
Models
To create a new version of your model, make necessary adjustments to your saved model file before exporting and store your exported model in Cloud Storage. Learn more
Name *
Name cannot be changed, is case sensitive, must start with a letter and may only contain letters, numbers and underscores. 0/128
Description
Python version *
Select the Python version that you used to train the model
Framework
Framework version
ML runtime version *
Machine type *
Single-core CPU
gs:// Model URI *
BROWSE
Enter the Google Cloud Storage path where you uploaded the model. Select a framework above for file format instructions.


Google Cloud Platform
kagenova ML development
AI platform
Version Details
Dashboard
AI Hub
Data Labelling
Notebooks
Jobs
Models
simple_test
Description
Model test_model
Model location gs://kagenova_ml_development/lecture/my_model/
Creation time 12 Feb 2020, 15:09:50
Last use time 12 Feb 2020, 15:49:27
Python version 3.7
Framework TensorFlow
Framework version 1.15.0
Runtime version 1.15
Machine type Single-core CPU
PERFORMANCE
EVALUATION BETA
TEST & USE
Test your model with sample input data
Request an online prediction by sending your input data instances as a JSON object.
Learn how to format input data
{
"instances": [
<value>|<simple/nested list>|<object>,
...
]
}

# Things to consider

- The model must be in the correct format (i.e. a save_model in TensorFlow)
- Limit on the size of the model (must be less than 256MB)
- Can scale to meet demand so great for the backend of a website
- Can create your own prediction code (known as custom prediction)
  - The code does not have access to the hard disk
- This should be seen a function in a microservice
  - The model should require as little pre or postprossing of the data as possible
  - People will add this to the model or use a custom prediction

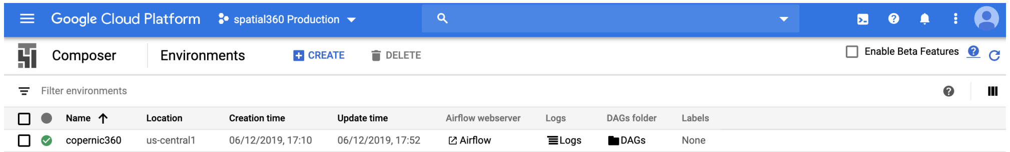
## Other versions

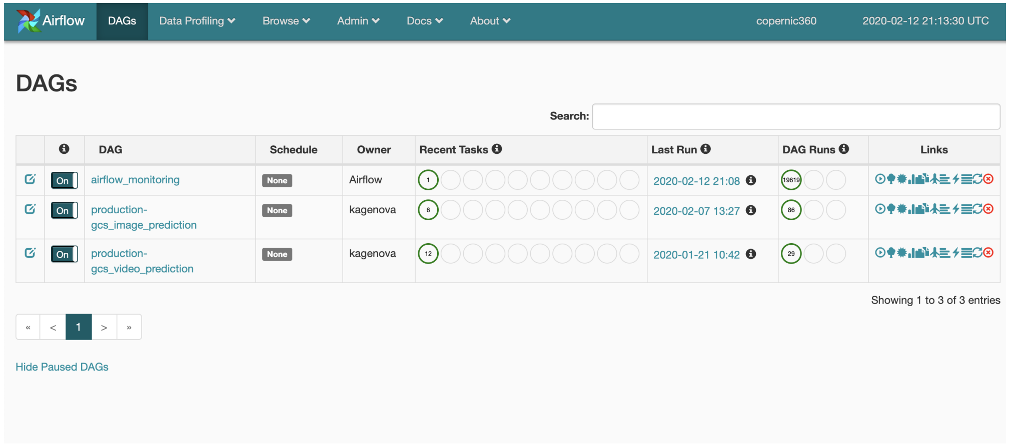
- AWS have Amazon SageMaker https://aws.amazon.com/machine-learning/ (https://aws.amazon.com/machine-learning/)

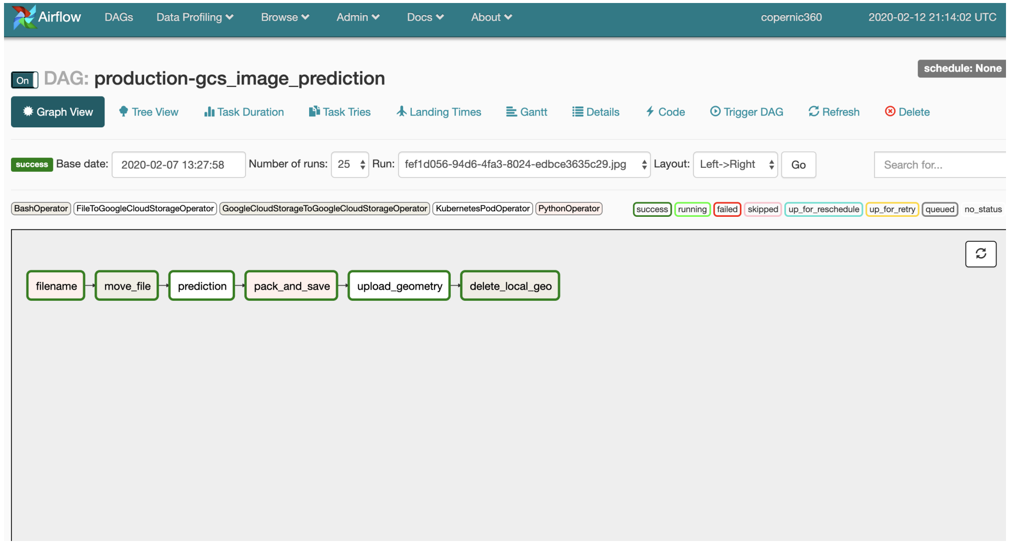
I need to perform regular data operations → **Airflow**

# I need to perform regular data operations → Airflow

- Build simple or complex data transform jobs
- Easy to
  - build
  - debug
  - monitor
- Best to use for regular jobs, e.g. once a day, minute etc.
- Can be used for one-off jobs (just not too often, as they can not be set off more then 10^6 per sec)
- See https://airflow.apache.org/ (https://airflow.apache.org/) and https://cloud.google.com/composer (https://cloud.google.com/composer)


Google Cloud Platform
spatial360 Production
Composer
Environments
CREATE
DELETE
Enable Beta Features
Filter environments
Name
Location
Creation time
Update time
Airflow webserver
Logs
DAGs folder
Labels
copernic360
us-central1
06/12/2019, 17:10
06/12/2019, 17:52
Airflow
Logs
DAGs
None


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
copernic360
2020-02-12 21:13:30 UTC
DAGs
Search:
DAG
Schedule
Owner
Recent Tasks
Last Run
DAG Runs
Links
On
airflow_monitoring
None
Airflow
2020-02-12 21:08
On
production-gcs_image_prediction
None
kagenova
2020-02-07 13:27
On
production-gcs_video_prediction
None
kagenova
2020-01-21 10:42
Showing 1 to 3 of 3 entries
Hide Paused DAGs




```python
    assembler = PythonOperator(
        task_id=f"assemble_predictions",
        python_callable=assemble_predictions,
        op_args=(
            "{{ti.xcom_pull('sampler')}}",
            *(f"{{{{ti.xcom_pull('prediction{i}')}}}}" for i in range(nworkers)),
        ),
    )

    save = PythonOperator(
        task_id=f"save",
        python_callable=save_params,
        op_args=(local_geo, "{{ti.xcom_pull('assemble_predictions')}}"),
        op_kwargs={"packer": pack_video, "version": params["6dof_version"]},
    )

    upload = FileToGoogleCloudStorageOperator(
        task_id="upload_geometry",
        src=local_geo,
        bucket=gcs_geo.bucket,
        dst=gcs_geo.blob,
    )

    delete_local_geo = BashOperator(
        task_id="delete_local_geo", bash_command=f"rm {local_geo}"
    )

    filename >> mover >> [
        detect_transitions,
        frame_rate,
    ] >> sampling >> key_frames >> predictions >> assembler >> save >> upload
    delete_local_geo << upload
    return dag
```

I need a simple front end for a website → Make a standard web app

# I need a simple front end for a website → Make a standard web app

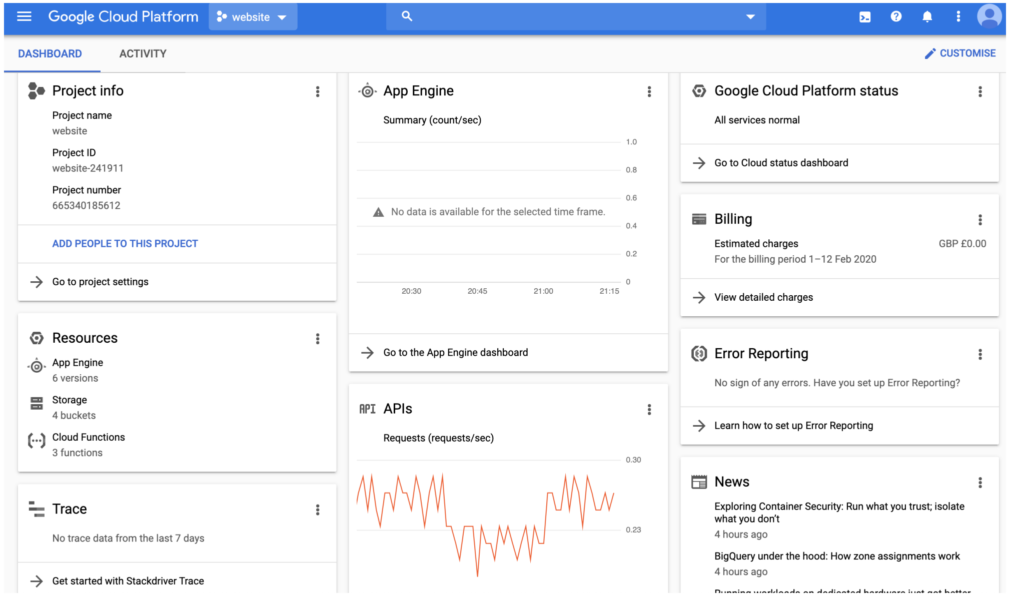
Virtual Machines are great but, as we saw with the AI platform, there can be easy to use set-ups in GCP.

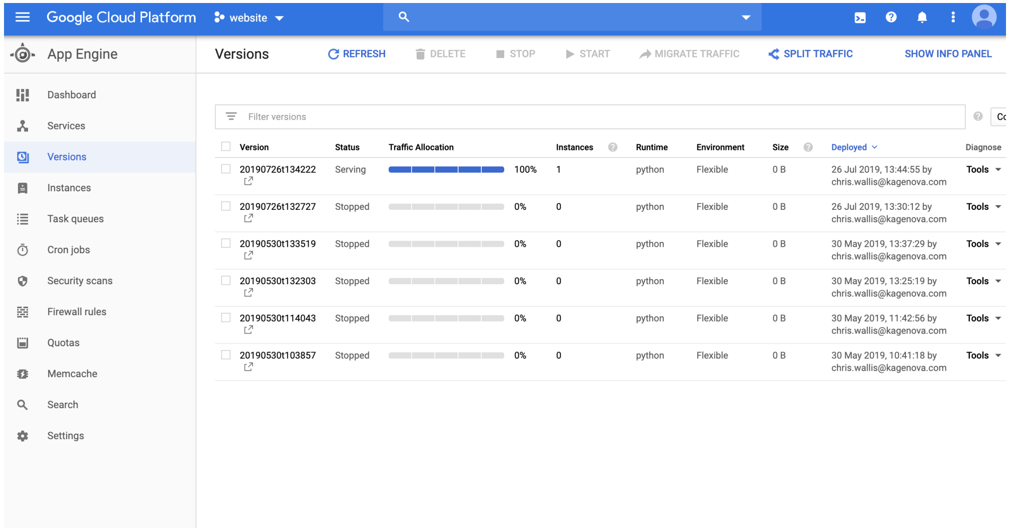
What do we need:

- The abilty to scale
- Connections to internet managed for us
  - URL is auto created
  - You can point a domain you own to the auto generated URL
- Simple way to write code
  - Every page or end point you want is simply a function you can write

We can use the Google App Engine (GAE) for this.

- See https://cloud.google.com/appengine (https://cloud.google.com/appengine)


Google Cloud Platform
website
DASHBOARD
ACTIVITY
CUSTOMISE
Project info
Project name
website
Project ID
website-241911
Project number
665340185612
ADD PEOPLE TO THIS PROJECT
Go to project settings
Resources
App Engine
6 versions
Storage
4 buckets
Cloud Functions
3 functions
Trace
No trace data from the last 7 days
Get started with Stackdriver Trace
App Engine
Summary (count/sec)
No data is available for the selected time frame.
1.0
0.8
0.6
0.4
0.2
0
20:30
20:45
21:00
21:15
Go to the App Engine dashboard
APIs
Requests (requests/sec)
0.30
0.23
Google Cloud Platform status
All services normal
Go to Cloud status dashboard
Billing
Estimated charges
GBP £0.00
For the billing period 1–12 Feb 2020
View detailed charges
Error Reporting
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting
News
Exploring Container Security: Run what you trust; isolate what you don't
4 hours ago
BigQuery under the hood: How zone assignments work
4 hours ago


Google Cloud Platform
website
App Engine
Versions
REFRESH
DELETE
STOP
START
MIGRATE TRAFFIC
SPLIT TRAFFIC
SHOW INFO PANEL
Dashboard
Services
Versions
Instances
Task queues
Cron jobs
Security scans
Firewall rules
Quotas
Memcache
Search
Settings
Filter versions
Version | Status | Traffic Allocation | Instances | Runtime | Environment | Size | Deployed | Diagnose
20190726t134222 | Serving | 100% | 1 | python | Flexible | 0 B | 26 Jul 2019, 13:44:55 by chris.wallis@kagenova.com | Tools
20190726t132727 | Stopped | 0% | 0 | python | Flexible | 0 B | 26 Jul 2019, 13:30:12 by chris.wallis@kagenova.com | Tools
20190530t133519 | Stopped | 0% | 0 | python | Flexible | 0 B | 30 May 2019, 13:37:29 by chris.wallis@kagenova.com | Tools
20190530t132303 | Stopped | 0% | 0 | python | Flexible | 0 B | 30 May 2019, 13:25:19 by chris.wallis@kagenova.com | Tools
20190530t114043 | Stopped | 0% | 0 | python | Flexible | 0 B | 30 May 2019, 11:42:56 by chris.wallis@kagenova.com | Tools
20190530t103857 | Stopped | 0% | 0 | python | Flexible | 0 B | 30 May 2019, 10:41:18 by chris.wallis@kagenova.com | Tools

## Flask

- Flask (https://www.fullstackpython.com/flask.html) is a simple web-app python package.
- Each part of a website is an endpoint.
  - Endpoints can take data (json or other universal formats)
  - They can return data or page (eg html code).
- Endpoints are created using a simple function decorator.
- See https://github.com/rmotr/example-flask-app/blob/master/rmotr/app.py (https://github.com/rmotr/example-flask-app/blob/master/rmotr/app.py) for a good simple example

```python
@app.route('/add', methods=['POST'])
def add_course():
    if not session.get('logged_in'):
        abort(401)
    name = request.form['name']
    instructor = request.form['instructor']
    description = request.form['description']
    if not all([name, instructor]):
        flash('ERROR. Missing data.', 'error')
    else:
        g.db.execute(
            'insert into courses (name, instructor, description) values (?, ?, ?)',
            [name, instructor, description])
        g.db.commit()
        flash('New entry was successfully posted')
    return redirect(url_for(DEFAULT_VIEW))
```

## Things to consider

- There is a Standard version:
  - The code cannot write or read from disk
  - Language and platform have more restrictions
  - Deployment is faster
  - Scaling is more responsive
- There is a Flexible version
  - Can have a much larger choice over what is deployed (using Docker to provision the VM)
  - Can read and write from disk
  - Deployment is slower
  - Scalling is less responsive
- Can read from GCS but this is slow (can lead to slow apps)
- Deployment is fast (a few tens of seconds to a couple of minutes)

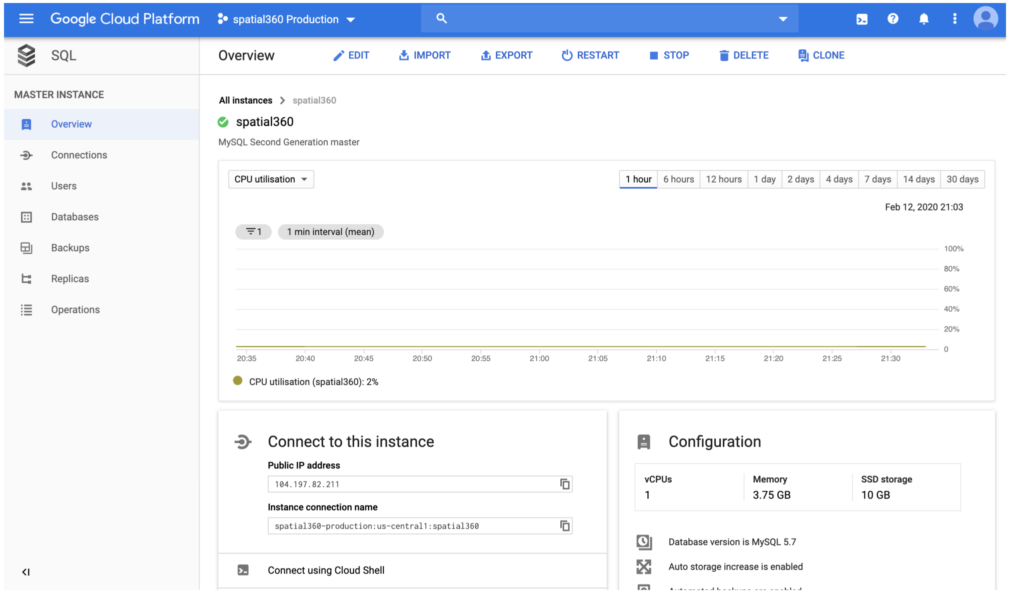
## Other versions

AWS haver product called Elastic Beanstalk https://aws.amazon.com/elasticbeanstalk/ (https://aws.amazon.com/elasticbeanstalk/). As GAE it allows you build web apps that can scale to meet the demand.

I need to have a database → SQL database

# I need to have a database → SQL database

- Database to store results from an experiment or users' data to a website or service
- Better to use Cloud SQL as all the connections and serving are done for you
- Can allow access through a simple UI or code to any account

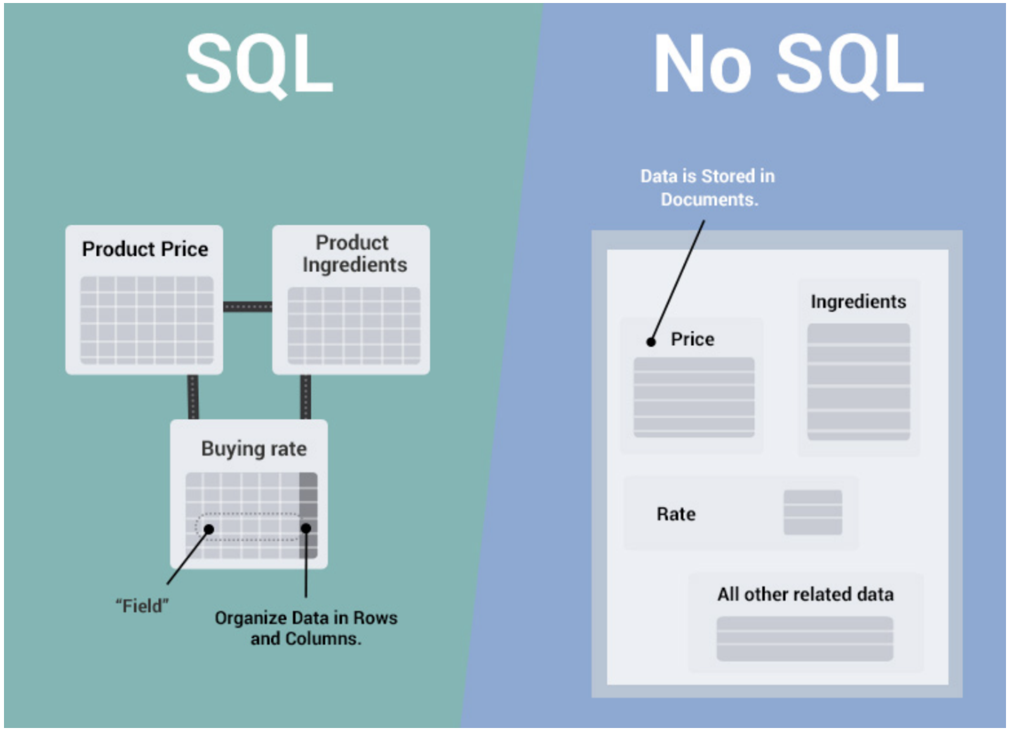

Google Cloud Platform
spatial360 Production
SQL
Overview
EDIT
IMPORT
EXPORT
RESTART
STOP
DELETE
CLONE
MASTER INSTANCE
Overview
Connections
Users
Databases
Backups
Replicas
Operations
All instances > spatial360
spatial360
MySQL Second Generation master
CPU utilisation
1 hour
6 hours
12 hours
1 day
2 days
4 days
7 days
14 days
30 days
Feb 12, 2020 21:03
1 min interval (mean)
100%
80%
60%
40%
20%
0
20:35
20:40
20:45
20:50
20:55
21:00
21:05
21:10
21:15
21:20
21:25
21:30
CPU utilisation (spatial360): 2%
Connect to this instance
Public IP address
104.197.82.211
Instance connection name
spatial360-production:us-central1:spatial360
Connect using Cloud Shell
Configuration
vCPUs
1
Memory
3.75 GB
SSD storage
10 GB
Database version is MySQL 5.7
Auto storage increase is enabled

# Things to consider

- There a lots of options for speed and reliable databases:
  - Cloud Spanner
  - Cloud SQL
  - Cloud Bigtable
  - Cloud Firestore
  - Firebase Realtime Database
  - Cloud Memorystore
- There could be a whole two lectures on the differences and benefits.
- Cloud SQL database is flexible and probably good for most use cases, unless scalabilty and speed is a must.
- The are two main differences relational or document based

# SQL or No SQL

- Tablular or Json data types



## SQL

### Relational Data Model

**Pros**

- Easy to use and setup.
- Universal, compatible with many tools.
- Good at high-performance workloads.
- Good at structure data.

**Cons**

- Time consuming to understand and design the structure of the database.
- Can be difficult to scale.

## No SQL

### Document Data Model

**Pros**

- No investment to design model.
- Rapid development cycles.
- In general faster than SQL.
- Runs well on the cloud.

**Cons**

- Unsuited for interconnected data.
- Technology still maturing.
- Can have slower response time.

# Other versions

AWS has Amazon RDS for SQL Server which seems to be the same

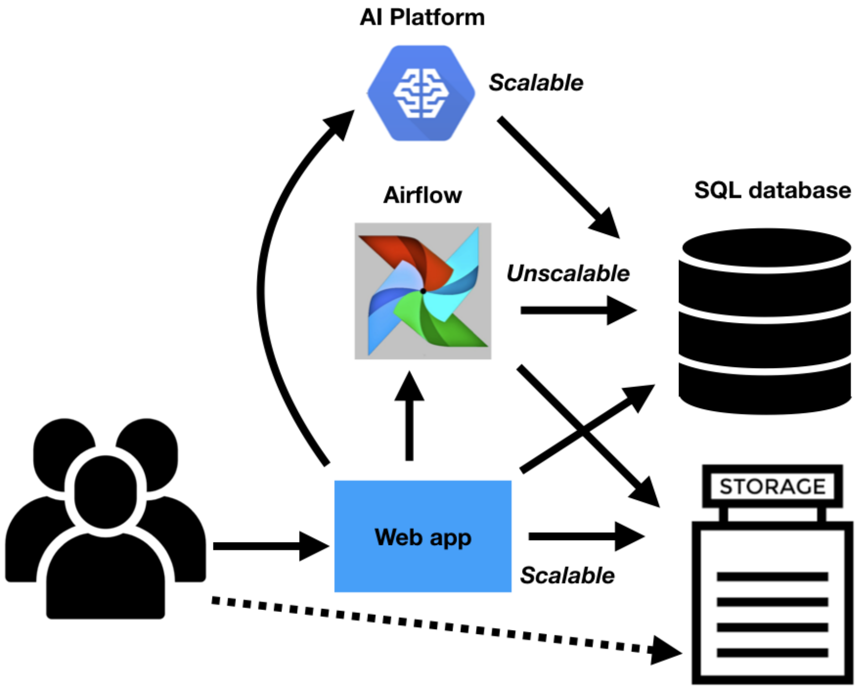
# Microservices

# Microservices

- You'll notice there are now many parts to our cloud computing tool box.

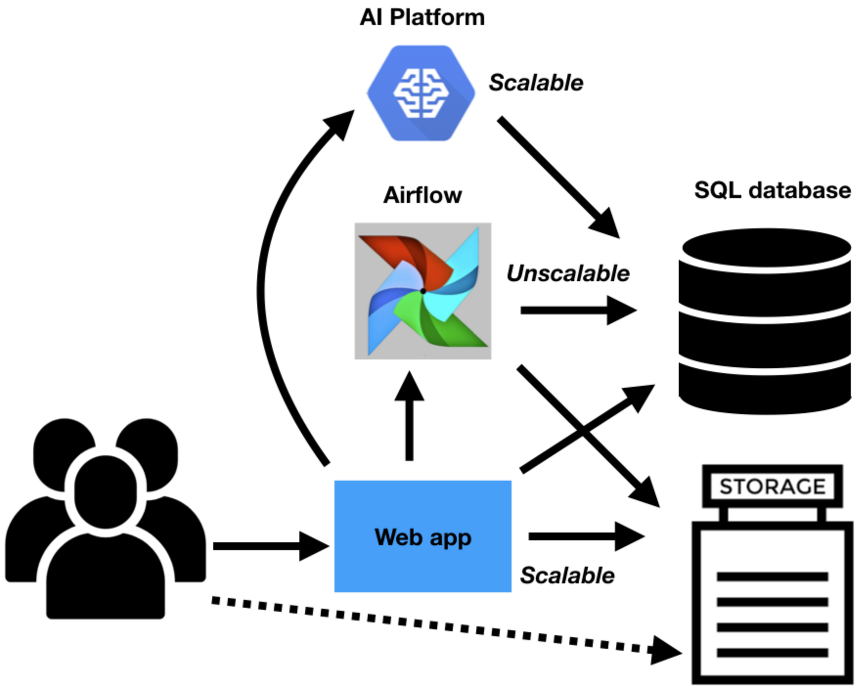
# Microservices

- You'll notice there are now many parts to our cloud computing tool box.



# Microservices

- You'll notice there are now many parts to our cloud computing tool box.

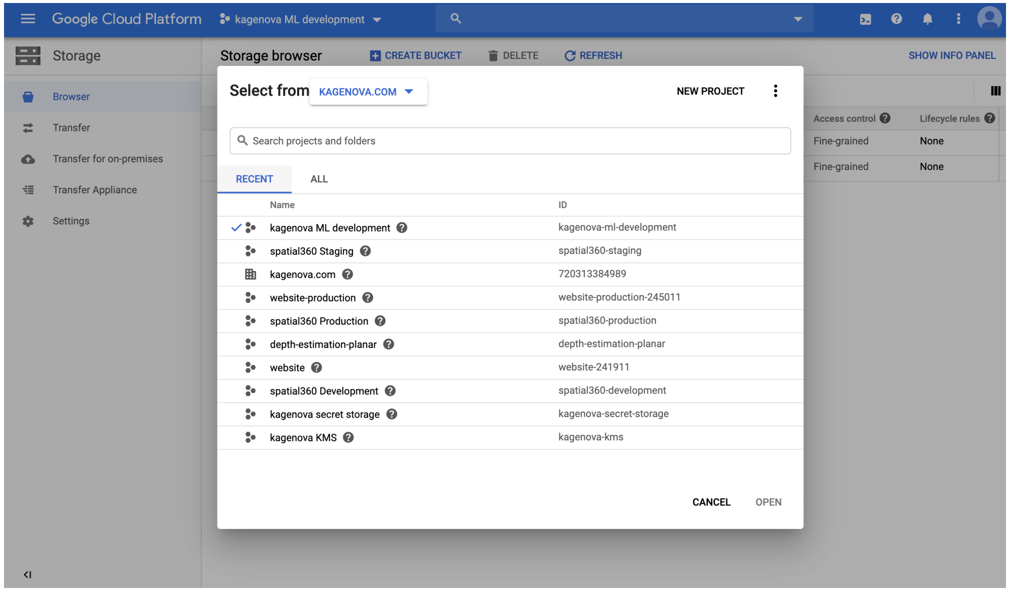


- In principle we could break the app up even further
    - Cloud function (GCP) and lamda functions (AWS) are endpoints that scale and perform simple tasks.
    - We could break some of the web app into many cloud functions then each would scale separately.
    - This way one endpoint can not effect any others (therefore much more stable)

I need to keep things seperate → Make good use of projects

# I need to keep things seperate → Make good use of projects

- As you can probably see in service there will be many microservice each performing their own tasks.
- In development we need to be able to try new versions
- This would be very risky if we didn't completly seperate **development** code from **production** code.


Google Cloud Platform
kagenova ML development
Storage
Storage browser
CREATE BUCKET
DELETE
REFRESH
SHOW INFO PANEL
Browser
Transfer
Transfer for on-premises
Transfer Appliance
Settings
Access control
Lifecycle rules
Fine-grained
None
Fine-grained
None
Select from
KAGENOVA.COM
NEW PROJECT
Search projects and folders
RECENT
ALL
Name
ID
kagenova ML development
kagenova-ml-development
spatial360 Staging
spatial360-staging
kagenova.com
720313384989
website-production
website-production-245011
spatial360 Production
spatial360-production
depth-estimation-planar
depth-estimation-planar
website
website-241911
spatial360 Development
spatial360-development
kagenova secret storage
kagenova-secret-storage
kagenova KMS
kagenova-kms
CANCEL
OPEN

## Things to consider

- You will need an automated way to deploy these services.
- Having a staging website allows easy viewing of new features in a website

## Other versions

AWS use different AWS Accounts to perform the same seperation.