Bringing the Energy with AWS

Deborah Aina, Nick Harbeck, Patrick Junghenn

Results

Below, we can see how a neural network can be used to predict the total energy consumption of a household, assuming amenities similar to the typical U.S. household.

Total Square Feet

4000

Total square feet that need cooling

1500

Total square feet that need heating

2000

Total number of AC units

4

○ Northeast ○ Midwest ○ South ◉ West

Update

Predicted Energy Consumption: 10692 KWH

Scope

Goal – Build a model in AWS that can predict the energy consumption of any U.S. residential home.

This project used Amazon Web Services to host data, train a neural network, and communicate findings via S3, EC2, and potentially other services.

Outcomes

1. Train a neural network on residential housing data to provide accurate predictions of energy consumption
2. Seamless integration among AWS services to allow real-time updates to the user interface
3. Tri-modal access to the machine learning for various audiences.

Features

TensorFlow provides the machine learning framework for training a neural network

AWS offers scalable opportunities for training and deploying models through EC2.

The user interface was developed as a website hosted on S3

Data

The U.S. Energy Information Administration (EIA) collects energy consumption data on buildings through its survey programs.

The 2015 Residential Energy Consumption Survey (RECS) collects information from over 5,600 households, covering hundreds of topics.

Architecture

Data Flow

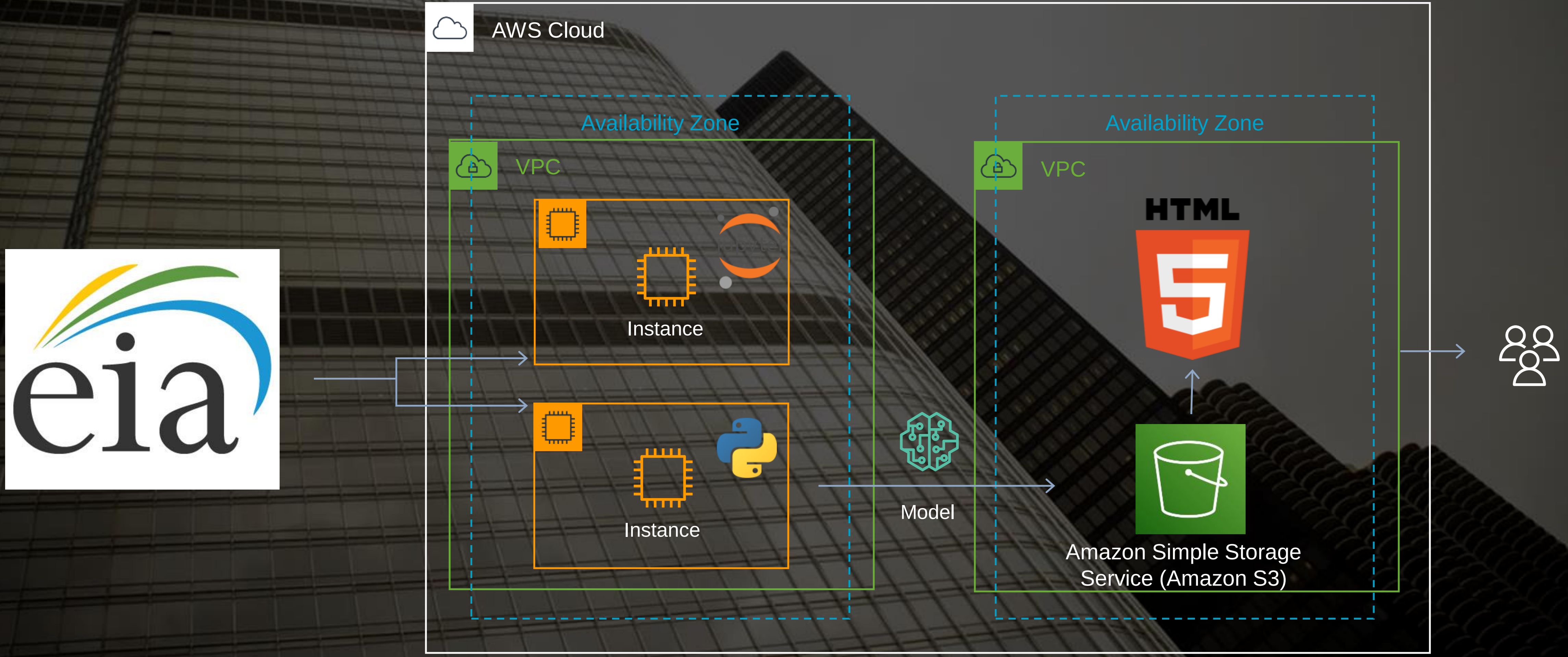
EC2 Instances

- Compute engine for Python-based training and model building
  - Trained models can interact with S3 to provide real-time updates to the user interface
- Compute engine for interactive Jupyter Notebook

S3 to host the website resources

- HTML pages for displaying the user interface
- JSON files holding the data and trained model
- TensorFlow.JS for TensorFlow model deployment, D3, HTML, CSS, JS

Architecture Diagram



Project Implementation

ec2-user@ip-172-31-81-139:~

```
[ec2-user@ip-172-31-81-139 ~]$ ls -l
total 16
drwxrwxr-x 2 ec2-user ec2-user     6 Apr 20 18:55 Notebooks
-rw-rw-r-- 1 ec2-user ec2-user 13761 Apr 20 19:13 RECS-CC-InteractiveNotebook.ipynb
[ec2-user@ip-172-31-81-139 ~]$
```

127.0.0.1:8888/notebooks/RECS-CC-InteractiveNotebook.ipynb

jupyter RECS-CC-InteractiveNotebook Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help Trusted Python 3

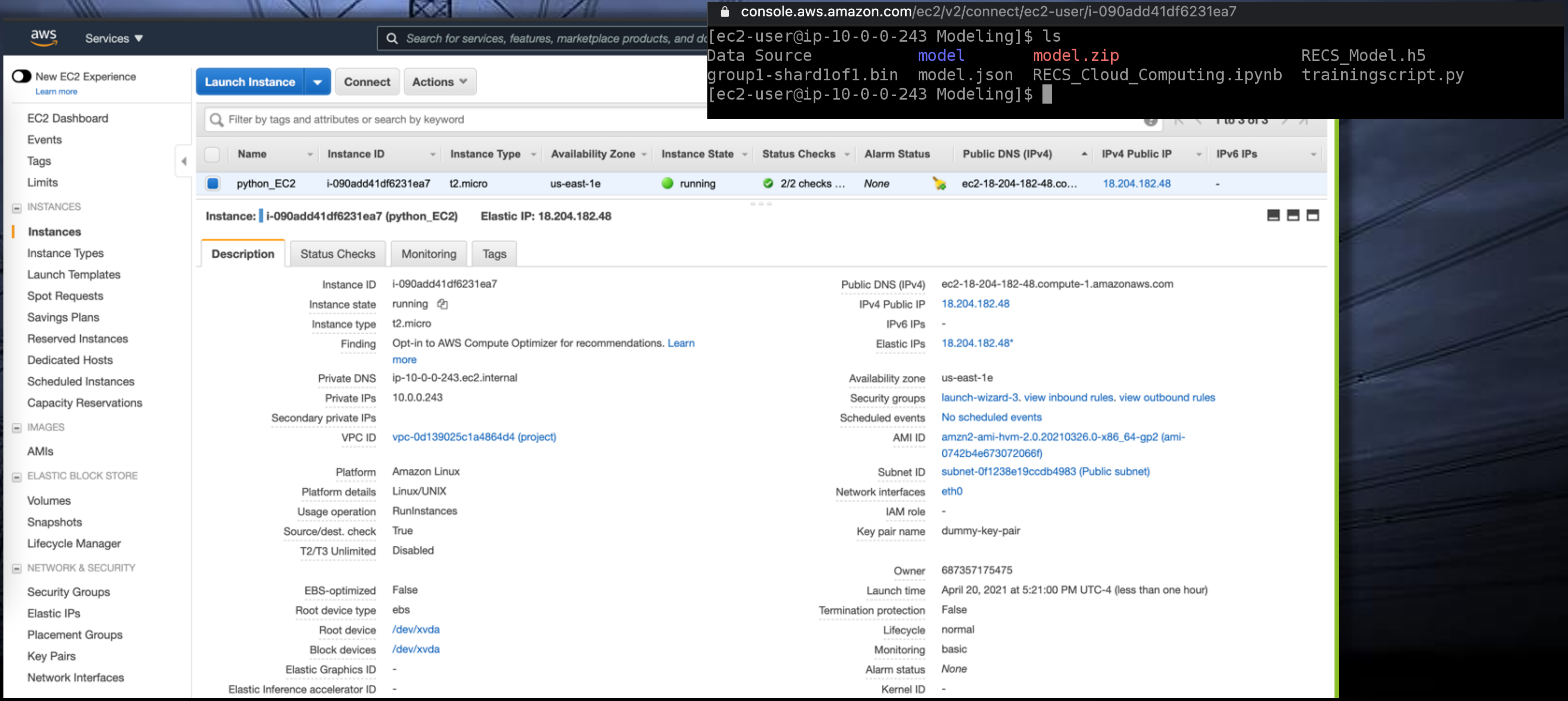
Run Code

In [7]:

```python
# evaluate the model
error_train = model.evaluate(X_train, y_train, verbose=0)
error = model.evaluate(X_test, y_test, verbose=0)

print('Training set MSE: %.3f' % error_train)
print('Training set RMSE: %.3f' % np.sqrt(error_train))
print('Test set MSE: %.3f' % error)
print('Test set RMSE: %.3f' % np.sqrt(error))

#Now that error is below the standard deviation, it looks like we have a working model!

#View how the test set predictions and true values vary
print("Average Absolute Error:", round(np.average(np.abs(model.predict(X_test)-np.array(y_test).reshape(-1,1))),3))
print("Error Standard Deviation:", round(np.std(model.predict(X_test)-np.array(y_test).reshape(-1,1)),3))

#Uncomment the below to see the results!
#print(np.array(model.predict(X_test)).reshape(-1,1)[:5])
#print(np.array(y_test).reshape(-1,1)[:5])
```

```
Training set MSE: 26397082.000
Training set RMSE: 5137.809
Test set MSE: 25434426.000
Test set RMSE: 5043.255
Average Absolute Error: 3550.263
Error Standard Deviation: 5042.566
```

Project Implementation



Project Implementation

- Using TensorFlow.js, we were able to convert the model output into a dense JavaScript model called into the website via the Content Delivery Network.
- Using S3, we load the static files, data and model unto AWS.
- The HTML, JavaScript, data and model files are successfully hosted in the same folder in an S3 bucket. The s3 endpoint is easily accessible by the public.
- Users can input overall square footage, square footage that needed heating, square footage that needed cooling and total number of A/C Units.

Future Expansion

- Connect the model output directly to the S3 folder called by the HTML page.
- Versioning allows access to different model timestamps. This is useful in keeping track of the model, HTML and JavaScript files.
- CloudFront could also be used to facilitate access in other parts of the world.

Demo and Questions