Open Cities AI Challenge: How We Scaled Building Segmentation

DSC 672 CAPSTONE SUPPLEMENT

JOHN HOFF

BRIAN PIRLOT

SEBASTIAN ZDAROWSKI

Scaling Model Training with Amazon AWS

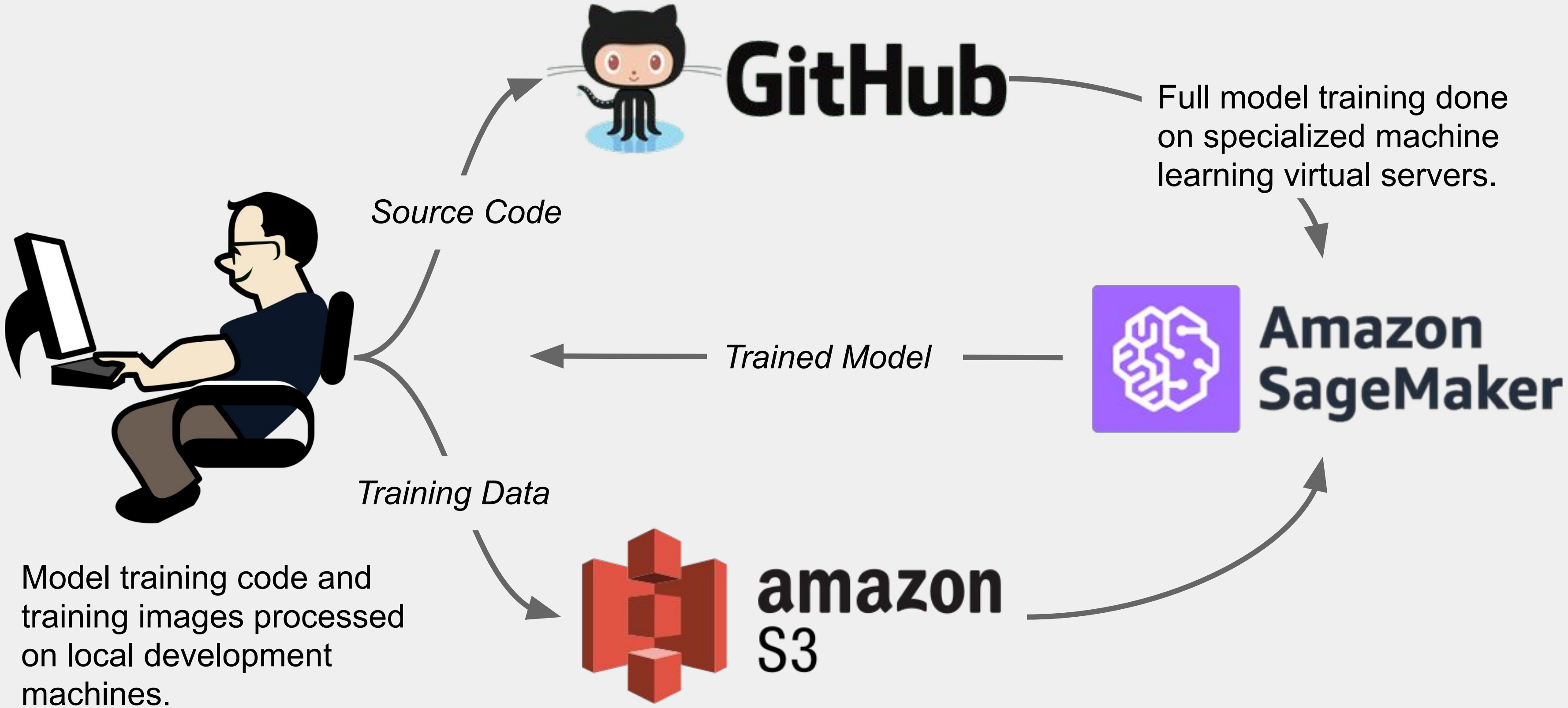
Amazon AWS Components

- SageMaker
  - Virtual GPU
- S3
  - Managing Training Data

Best practices that make this approach work

- Source Control
  - Git
- Python Virtual Environments
  - virtualenv
  - conda

Let's Take a Closer Look at Our Workflow



Full model training done on specialized machine learning virtual servers.

Model training code and training images processed on local development machines.

Best Practices for Python-Based ML

I consider the following best practice a hard requirement before attempting to scale with on-demand cloud resources. Adding remote hardware is equivalent of adding a remote team member - and even a team of one now needs to collaborate.

Source Control
Manage changing code with industry-leading software engineering tools that is built distribution of code as the primary workflow.

Python Virtual Environments
Manage the versions and dependencies of Python libraries making whole environments reproducible.

Source Control - Git

Git is a free and open source distributed version control system.

- Application code to do data processing, model training, and analysis
- Configuration files for tools and environments
- Data is usually not included

Software

- Git (Terminal)
- GitHub Desktop
- Source Tree
- TortoiseGit
- Most IDEs

Service Providers

- GitHub
- Bitbucket
- GitLab

Python Virtual Environments

The Python interpreter and standard library are only a fraction of the dependencies of a Python-based data science project.

To training a Solaris model in our project depends on 120 different Python modules weighing in at almost 2 GB. These modules also depend on specific versions of the other modules. Managing this by hand is inviting problems. Sharing this environment by hand is virtually impossible.

The standard Python distribution weighs in at less than 100 MB...just 5% of the overall project dependencies.

Python Virtual Environments - PIP style

What you typically see:

- **venv**: directory containing a sandboxed version of python and all modules.
- **requirements.txt**: file containing all dependencies to install and the indicated version.

pip
The Python Package Installer
A set of cross platform tools and module indexes to enable the easy installation and management of Python modules.

Core audience is the overall python community

Python Virtual Environments - CONDA Style

What you typically see:

- **environment.yml**: file containing all dependencies to install and the indicated version.

Conda is used to manage both the Python modules *and* the Python interpreter and distribution.

conda
Anaconda and Miniconda

Open source package management system and environment for any language

Core audience is the data science community

Thanks For Your Patience

Now that you are properly wrangling your code, you are ready to dive into Amazon AWS. I will be opening up each section in AWS to demonstrate the functionality and how it was used in our project.

We will cover:

- AWS Console
- AWS Cost Explorer
- S3
- Amazon Sagemaker
- JupyterLab

AWS Console

The AWS console is essentially a “big bag of services”.

- Find the services you need
- How to bookmark services to streamline the experience

The “big bag of services” is a theme Amazon extends to their command line interface. Almost every action taken in the web UI can be accomplished using the CLI in a shell...a singular application with a dizzying array of argument options.

AWS Cost Explorer - Billing Interface

Before you start running any service you need to familiarize yourself with how you are going to be charged by Amazon.

- Service Breakdowns
- Budgets
- Alerts

I cannot stress enough that the first step to using AWS should be to create a billing alert. You do not want to be in the position of requesting forgiveness for charges.

S3 - Blob Storage

One of the oldest AWS services, S3 makes it simple and cost effective to store and share files.

Software for S3

- AWS CLI
- S3 Browser
- Dragon Disk

Some machine learning tasks will use S3 directly for data and model storage. This was skipped for our project to keep things simple. We used S3 for sharing between ourselves and SageMaker.

Amazon SageMaker - ML Optimized Computing

Virtual servers specialized for machine learning. Slightly more expensive, but Amazon manages all of the drivers and dependencies so you don’t have to.

- Dedicated GPUs available
- High memory instances
- On demand usage

Using Amazon SageMaker - JupyterLab

Brings everything to the table that was missing from Jupyter notebook when running on remote machines. Far more feature-rich that typically notebook “colabs” as it isn’t trying to abstract away the underlying hardware.

- Integrated Notebook
- Local terminal shells
- Local file explorer
- Git client

Suggested Services to Review

This presentation has assumed you will be the sole user of the Amazon infrastructure. Once you are ready to start working with multiple users and or projects here is where you will start:

- IAM
  - User and Permissions Management
- Resource Groups
  - Logically grouping and isolating services

Where to Now?

Feel free to explore our capstone project to see how everything has been packaged and organized. I have done a decent job of documenting everything.

https://github.com/theBraindonor/DSC672
https://git.io/JvKPZ

Feel free to contact me and stay in touch:
john.hoff@braindonor.net
jhoff@productiveedge.com

Recommended Resources

- An Introduction to Git for Data Scientists
  https://www.datacamp.com/courses/introduction-to-git-for-data-science
- A Guide to Python Virtual Environments
  https://towardsdatascience.com/virtual-environments-104c62d48c54
- Build, Train, and Deploy a Machine Learning Model with Amazon Sagemaker
  https://aws.amazon.com/getting-started/tutorials/build-train-deploy-machine-learning-model-sagemaker/
- Getting Started With JupyterLab
  https://dzone.com/articles/getting-started-with-jupyterlab