APPENDIX E
Building a Technical Portfolio for MLOps

By Noah Gift

Something I recommend to everybody is to become a "Triple Threat." In basketball, a triple threat position means someone who can score, rebound, or pass. The defender has to guard against all three options. You can also be a triple threat from a career perspective, meaning you can get past the hiring gatekeepers because you have three threats: a technical portfolio, a relevant certification, and either work experience or a relevant degree.

I advise on Cloud Certifications in Appendix B, but here we're going to dive into your technology portfolio. Building a portfolio project builds up your resume, which will have a link to high-quality source code and a demo video, versus another resume that does not. Who would you call in for an interview? I would call in the candidate who already showed me what they could do before we met.

In building a project, pick something you think will "move the needle" in the next two years. Consider the following ideas:

- GitHub project with source code and a *README.md* explaining the project. The *README.md* should be of professional quality and use a business writing style.
- 100% repeatable notebooks or source code.
- Original work (not a copy of a Kaggle project).
- Authentic passion.
- Five-minute final demo video showing how it works.
- The demo needs to be very technical and show precisely how to accomplish a task, i.e., you need to teach someone step by step. (Think about a cooking show where a chef demonstrates how to make chocolate chip cookies. This level of detail needs to be similar.)

- The video should be at least 1080p with a 16:9 aspect ratio.
- Consider recording with a low-cost external mic.

Something to consider in building a Jupyter notebook is to break down the steps in a data science project. Typically the steps are:

- Ingest
- EDA (Exploratory Data Analysis)
- Modeling
- Conclusion

For MLOps portfolio projects, here are a few ideas that have resulted in students getting jobs at major tech companies, i.e., FAANG, and cutting-edge startups as ML engineering, data engineers, and data scientists.

Project: Continuous Delivery of Flask/FastAPI Data Engineering API on a PaaS Platform

An excellent way to test your knowledge of building APIs is the following project:

1. Create a Flask or Fast application on a cloud platform and push source code to GitHub.
2. Configure a cloud native build server (AWS App Runner, AWS Code Build, etc.,) to deploy changes to GitHub.
3. Create a realistic data engineering API.

You can refer to this O'Reilly walkthrough of using functions with FastAPI to build Microservices for more ideas, or this sample Github project for a complete working AWS App Runner starter project for FastAPI.

Project: Docker and Kubernetes Container Project

Many cloud solutions involve Docker format containers. Let's leverage Docker format containers in the following project:

1. Create a customized Docker container from the current version of Python that deploys a Python ML application.
2. Push the image to DockerHub, Amazon ECR, Google Container Registry, or some other Cloud Container Registry.
3. Pull the image down and run it on a cloud platform cloud shell: Google Cloud Shell or AWS Cloud9.

4. Deploy an application to a cloud-managed Kubernetes cluster, like GKE (Google Kubernetes Engine), or EKS (Elastic Kubernetes Service), etc.

Project: Serverless AI Data Engineering Pipeline

Reproduce the architecture of the example serverless data engineering project shown in Figure E-1. Then, enhance the project by extending the functionality of the NLP analysis: adding entity extraction, key phrase extraction, or some other NLP feature.

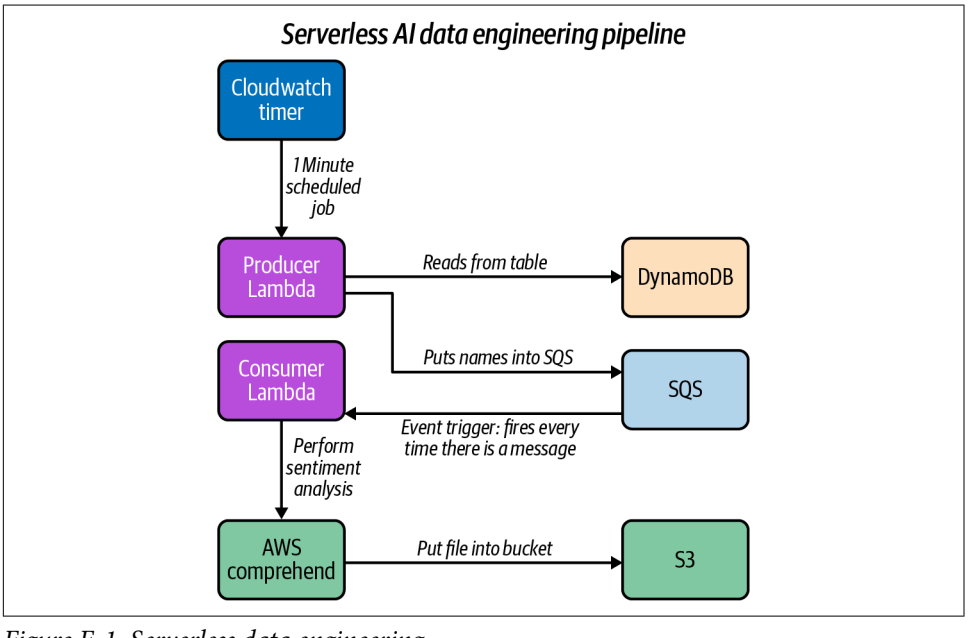

Figure E-1. Serverless data engineering

A good resource for this project is the following repo: *https://github.com/noahgift/awslambda*.

Project: Build Edge ML Solution

Build and deploy an edge-based computer vision solution using one of the technologies covered in the book:

- Intel Movidius Neural Compute Stick 2
- AWS DeepLens
- Coral AI
- SmartPhone (iOS, Android)

- Raspberry Pi

Here are the deliverables for your project:

- Publish your results as a Jupyter notebook, a folder of Colab notebooks, or both in a GitHub repo.
- Write two-page synopsis of a research project in PDF format.
- Include a link to your published works in your two-page synopsis.
- Create a 60-second demo video of your working computer vision project making inference (predictions).
- Submit your project to Intel or AWS Community Projects.

Project: Build Cloud Native ML Application or API

Build a cloud native analytics application hosted on the Google Cloud Platform (GCP), AWS, Azure, or another cloud or technology (i.e., Kubernetes). This project aims to give you the ability to create realistic, working solutions that work with modern techniques.

Before you begin, read "Hidden Technical Debt in Machine Learning Systems" by Sculley et al. (2015).

A good idea to limit complexity is to use a public dataset. One example could be a project using public data from a Google BigQuery dataset if using GCP. Alternately, if using AutoML, data can be tutorial data or custom data.

The main idea is for you to think about starting to create a portfolio. Here is a list of suggested project requirements for a GCP project; you can modify the tech stack for a different cloud.

- Source code stored in GitHub
- Continuous deployment from CircleCI
- Data stored in GCP (BigQuery, Google Cloud Storage, etc.)
- ML predictions created and served out (AutoML, BigQuery, etc.)
- Stackdriver installed for monitoring
- Google App Engine serves out HTTP requests via REST API with a JSON payload
- Deployed into GCP environment
- A two-page, single-spaced paper describing the project as a consultant would describe it to a client during the hand-off phase

The following is an example of a final project checklist to consider.

- Does it perform ML prediction/inference?
- Are there separate environments?
- Is there comprehensive monitoring and alerts?
- Is the correct datastore used, i.e., relational, graph, key/value?
- Does the principle of least privilege apply?
- Is the data encrypted in transit and at rest?
- Did you load-test the application to verify performance?

Getting a Job: Don't Storm the Castle, Walk in the Backdoor

The technology industry always has a "dream" job title. These titles come and go. Here are a few: Unix systems administrator, network administrator, webmaster, web developer, mobile developer, data scientist. What happens when these job titles appear is that companies panic about hiring these positions.

Then all progress grinds to a halt, and more and more hoops appear. A classic example is asking someone for ten years of experience on a technology when it has been around for one year. Finally, it becomes impossible to get into "the castle." The front of the castle has guards, hot oil, spears, and a monster waiting in the moat. Instead, think about a backdoor. Often a backdoor is a less prestigious job title.

How to learn
: Software-related careers are unique compared to other professions. Software-related jobs have much more in common with professional athletes, martial artists, or musicians. The process of achieving mastery involves an embrace of suffering and a love of making mistakes.

Create your own 20% time
: Never trust a company to be the sole source of what you need to learn. You have to carve out your learning pathway. One way to do this is to spend a couple of hours a day learning new technology as a habit. Think about it as an exercise for your career.

Embrace the mistake mindset
: It is common to avoid mistakes and want perfection. For example, we try to avoid car accidents, dropped groceries, and other mistakes, and most students want to get a perfect "A" on an exam.

  In learning to be a competent software engineer, it is better to flip this on its head. A constant stream of mistakes means you are on the right track. How many

mistakes have you made this week? How many have you made today? William Blake said this best in 1790 when he said, "If the fool persisted in his folly, he would become wise."

Finding parallel hobbies to test your learning

If you asked a quorum of successful software engineers with 10+ years of experience, you would hear some version of this: "I am a learning machine." So then, how does a learning machine get better at learning? One method is to pick a sport that takes years to master that you are a complete beginner in and observe yourself.

Two games in particular that are well suited for this activity are rock climbing and Brazilian Jiu-Jitsu. Brazilian Jiu-Jitsu has the added side-effect of teaching you practical self-defense. You will find out that you have blind spots in learning that you can then fix in your "real" job.