Learn AWS Amplify

The Basics

AWS Amplify is a command-line utility that generates web and mobile applications on Amazon Web Services (AWS) servers. It is written in node and requires NodeJS. My recommendation is to use nvm to install NodeJS if you're installing NodeJS for the first time.

Use Cases

- You posted your app idea on Craigslist, but no developers are willing to work for free, so you'll do it yourself!
- You want to create a proof of concept application quickly.
- You would like to evaluate several different technologies without having to learn the internals.
- You want to do something more than basic form collection.

You might want to evaluate Honeycode if you're doing basic data collection from web and mobile platforms.

Installation

The latest and greatest installation instructions for a react application work well from beginning to end. The last time I tried to learn AWS Amplify and follow the AWS Amplify instructions, it ended in a whole lot of errors. This time around, everything worked as expected.

Amplify Choices

When creating an amplify application you can build either a web or a mobile application. In the case of the mobile applications you can base your apps on web frameworks like React and Ionic but also gain the ability to manipulate system level APIs.

Amplify Console

Running this command pops you into the AWS console.

```
amplify console
```

I've found that clicking on the app name after running this command is the most useful route. I'm able to get to the build status of the project since I decided to automated my deployment with a GitHub repository.

Push

If you’re looking to update your API you’ll want to push up the changes with this command.

```
amplify push
```

Update

If you’ve made changes to your schema you’ll want to update your API.

Run this command to push the changes and generate your schema.

```
amplify update api
```

Wrapping Up

I wrote a short article because there isn’t a lot to explain with AWS Amplify at a high level. It can get complicated when you’re working to keep your resources in order. If you’re not looking to do something complicated and want to do some data collection with a few bells and whistles, AWS Amplify will fit the bill.

Learn AWS Amplify – Beyond the Basics

So far there aren’t many resources for learning AWS Amplify but here are a few.

- AWS Amplify: Console User Guide
- Learning AWS Amplify (LinkedIn)
- Serverless React with AWS Amplify – The Complete Guide (Udemy)
- AWS AppSync & Amplify with React & GraphQL – Complete Guide (Udemy)

Learn EC2

The Basics

Learn EC2 basics, concepts, and more.

Amazon EC2 is the core compute service provided by Amazon Web Services. It’s Virtual Machines in the cloud.

If you have been using Amazon Web Services for a while there is no doubt you encountered EC2 in one way or another. It is a central service used by Amazon Lightsail and numerous other offerings.

Primary Use Cases

1. Hosting environments for your apps
2. High-performance computing
3. Computing with more specific use cases like:
    1. GPU Heavy Tasks
    2. High Memory Requirements

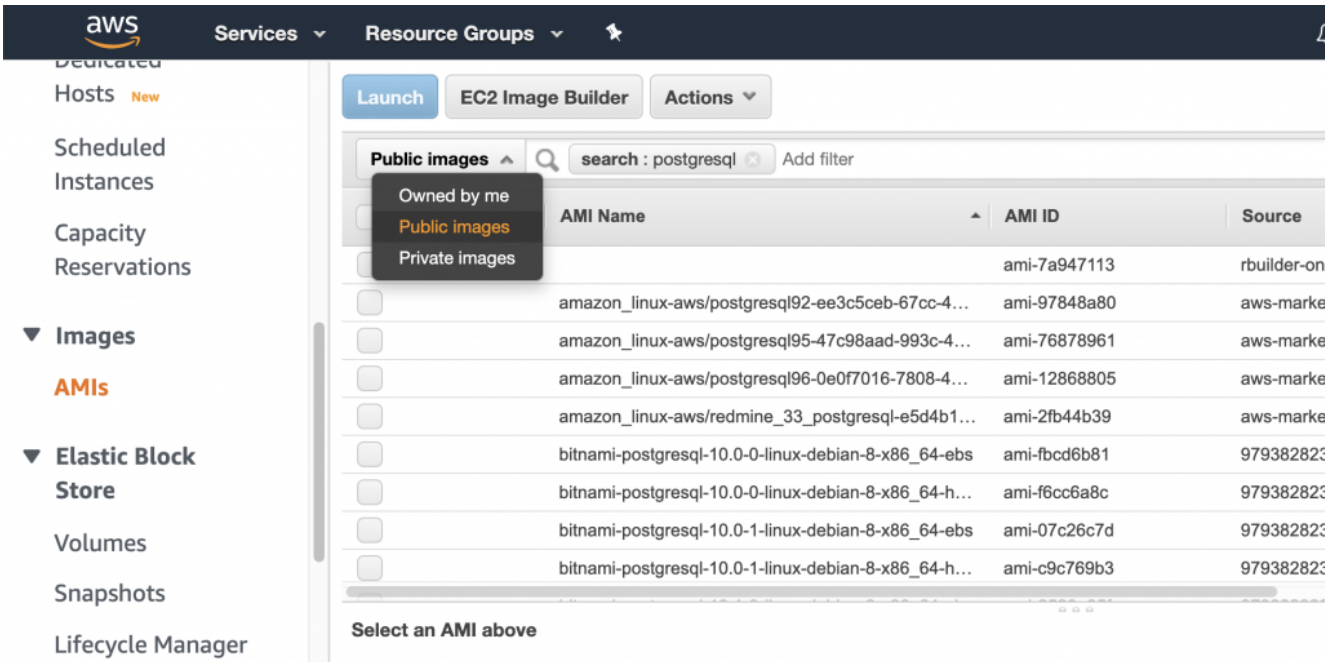
3. Burstable compute

Instances

EC2 Instances are central to the ecosystem. To use EC2, you will launch an EC2 instance from an EC2 image. An EC2 image contains a snapshot of an operating system like Linux with everything configured as it was upon snapshot creation.

Images

Amazon Machine Images (AMI) are prepackaged in numerous ways to meet a legion of use cases. Public images are found within the AWS console's AMI search feature. If you have an AWS account, you can access the search feature.



search : postgresql

AMI Name	AMI ID	Source
	ami-7a947113	rbuilder-on
amazon_linux-aws/postgresql92-ee3c5ceb-67cc-4…	ami-97848a80	aws-marke
amazon_linux-aws/postgresql95-47c98aad-993c-4…	ami-76878961	aws-marke
amazon_linux-aws/postgresql96-0e0f7016-7808-4…	ami-12868805	aws-marke
amazon_linux-aws/redmine_33_postgresql-e5d4b1…	ami-2fb44b39	aws-marke
bitnami-postgresql-10.0-0-linux-debian-8-x86_64-ebs	ami-fbcd6b81	979382823
bitnami-postgresql-10.0-0-linux-debian-8-x86_64-h…	ami-f6cc6a8c	979382823
bitnami-postgresql-10.0-1-linux-debian-8-x86_64-ebs	ami-07c26c7d	979382823
bitnami-postgresql-10.0-1-linux-debian-8-x86_64-h…	ami-c9c769b3	979382823

Select an AMI above

Security Groups

Security groups are a collection of firewall rules that secure your instances from nefarious hackers. If you're familiar with firewalls, then you will be familiar with security groups. Apply the principle of least privilege and only grant access to systems that require access to your EC2 instances.

Other Concepts

Learn EC2 concepts you'll need to understand when using EC2.

Key Pairs

Use key pairs to gain secure shell access on your EC2 instance.

Volumes

Attach disks to your EC2 instances to expand space and meet various use cases like high performance read/write disk access.

Load Balancers

Scale up your instances by routing traffic to multiple instances.

Auto Scaling Groups

A definition of how many instances you want to run within a cluster along with thresholds on when to expand to the size of the cluster.

Snapshots

Create a moment in time snapshot of your EC2 instances.

Capacity Reservations

Save money by reserving your EC2 instances in advance.

Learn EC2 – Beyond the Basics

- **Books**
  - Programming Amazon Ec2 (O'Reilly)
- **Videos**
  - Amazon EC2 Fundamentals (LinkedIn)
  - Managing AWS EC2 Instances (Pluralsight)

Related Content

Cloud providers

The Basics

Cloud providers have taken the tech world by storm over the past several years. Amazon led the way with cloud computing in 2006 when it launched the Amazon Elastic Compute Cloud (EC2) service. The cloud, however, was mentioned as early as 1998.

The Flexera state of the cloud report provides a snapshot of the current state of cloud computing. The Flexera report is compiled once a year and contains insights about how companies are using cloud providers, along with adoption rates.

Primary Use Cases

Companies use cloud providers for numerous reasons. There are many use cases due to the flexibility and interoperability cloud services provide.

Here are some of the primary use cases.

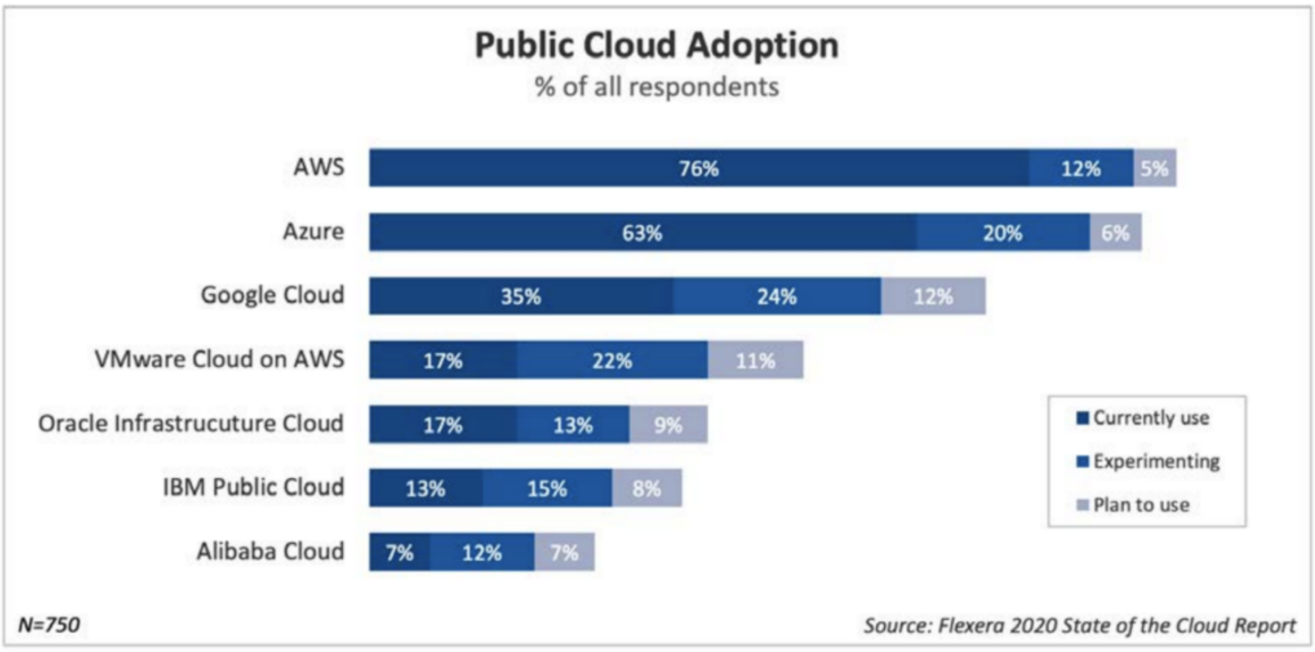
- Hosting servers on virtual machines and containers
- Solving large scale computing problems like big data analysis, machine learning, and artificial intelligence.

Adoption Rates

Cloud providers have enjoyed massive growth due to rapid adoption amongst companies going through digital transformations. Cloud adoption is mixed between public and private clouds.

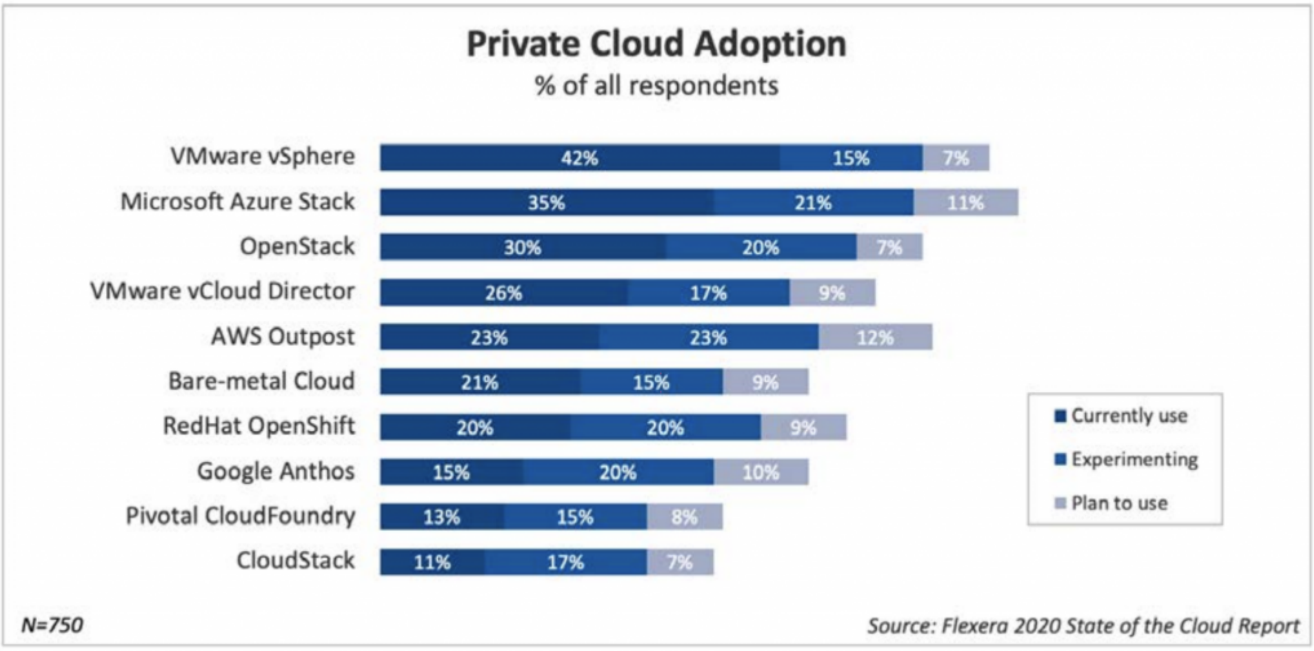
Public Cloud Adoption

Amazon Web Services leads the pack while Microsoft Azure is not far behind. Google has a fair share but lags due to a late start and a reduced number of relationships with large enterprises.



Private Cloud Adoption

Private clouds are an alternative to public clouds. These services allow people to reduce the complexity of hosting varied resources into a single-entry point of APIs, much like AWS, Azure, and other public cloud providers.



Amazon Web Services

In 2002 the AWS platform was born. In my eyes, this move was a classic case of dogfooding where Amazon needed web scale to deliver their shopping experience at scale. Why wouldn't Amazon attempt to recoup some of their hosting costs by reselling their platform?

The AWS platform applies the single-responsibility principle to its services in most cases. Alternatively, services like LightSail aggregate services into a single service.

Microsoft Azure

Microsoft Azure is like AWS in many ways. It provides scalable services that complement their business productivity solutions. Azure’s cloud services tend to appear like a monolith within the Azure Web Portal. Under the hood, there are single responsibility services like Azure Functions that allow you to run functions in the cloud, much like AWS with AWS Lambda.

Google Cloud

Google’s services have evolved in fits over the years. Google App Engine was released in 2008 and was meant solely for Python applications with its first iteration. Fast forward to the present, and Google App Engine now supports a multitude of languages. Google’s cloud services bear a resemblance to other cloud services, as evidenced by the Google Cloud Functions service.

The Rest

I could go on and on like Larry Ellison, but I’ll spare you that. Luckily, the software industry is a shining bastion of commerce. There is plenty of room for the likes of Oracle, VMWare, Red hat, and others. The industry is booming, and competition is fierce just the way I like it.

Conclusion

Cloud providers are a continually evolving beast. Trends come and go, and the war rages on between cloud-native vs. containers. Developers are the clear winner of the cloud computing wars. If we can keep choice overload at bay, we will continue to benefit from the power these platforms provide.

Cloud Providers – Beyond the Basics

- **Books**
  - Digital Transformation: Survive and Thrive in an Era of Mass Extinction (Amazon)
  - A Brief Guide to Cloud Computing (Amazon)
- **Videos**
  - Fundamentals of Cloud Computing (Pluralsight)
  - Cloud Computing: The Big Picture (Pluralsight)
  - Learning Cloud Computing: Core Concepts (LinkedIn)
  - Cloud Computing Concepts (Udemy)

Related Content

Learn Amazon Athena

The Basics

Learn Amazon Athena basics. Amazon Athena is meant for querying copious amounts of data on the cheap. It’s a straightforward service based on [Presto]() with some interesting integrations. The Presto Foundation calls Amazon Athena and Amazon EMR [Presto Cloud on their website.]()

It’s not dirt cheap to use Amazon Athena, but it is convenient. It costs $5.00 per TB of data scanned. It’s possible to pay a third of that with compression. Additional savings occur when querying a single column. The big win with Athena is the cluster of servers that AWS has at the ready. It wouldn’t be straightforward to set up your own Presto analytics engine.

Primary Use Cases

- Querying vast amounts of
  - Log Data
  - Behavioral Data
  - Noncritical data

Importing Data

I’m facing a scenario where I need to query a 58G CSV file. I was able to load the ~1 billion rows into PostgreSQL, but any queries on the data caused my work machine to crash. I wrote a little python script to query the data which is working but is taking too long to process. So, I brought out the big guns.

I decided to try Athena out as a solution to my problem. Here’s how it went down.

I created this table to store my CSV file. This table is for storing the output of an AWS command-line query.

```
DROP TABLE learnamazonathena.s3_objects;
CREATE EXTERNAL TABLE IF NOT EXISTS
learnamazonathena.s3_objects (
         time string,
         bytes bigint,
         object string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
STORED AS TEXTFILE
LOCATION 's3://learn-amazon-athena/csv';
```

Amazon S3 is now the datastore for our CSV file. We generate this CSV file with these commands.

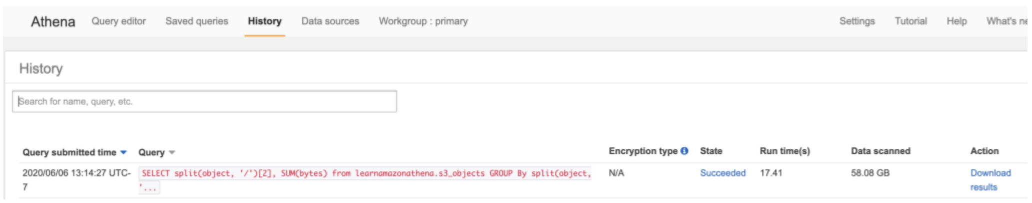
```
# Query all S3 objects recursively
aws s3 ls --recursive learn-amazon-athena > s3_objects.csv

# Format the output as a CSV file
perl -p -i -e
's/(^.{0,19})(\s+)([0-9]+)(\s)(.+)$/"\1","\3","\5"/g'
applications.csv
```

I then ran this query against the newly created table.

```
SELECT split(object,
         '/')[2], SUM(bytes)
FROM learnamazonathena.s3_objects
GROUP BY  split(object, '/')[2]
```

This query took seventeen seconds to complete.



Query submitted time	Query	Encryption type	State	Run time(s)	Data scanned	Action
2020/06/06 13:14:27 UTC-7	`SELECT split(object, '/')[2], SUM(bytes) from learnamazonathena.s3_objects GROUP By split(object, '...`	N/A	Succeeded	17.41	58.08 GB	Download results

I love how fast and straightforward Amazon Athena is. The CSV file I created is small in the grand scheme of things. Facebook created Presto to query obscene amounts of data then later open-sourced it.

Learn Amazon Athena – Beyond the Basics

- **Print**
  - Getting Started – User Guide (AWS)
  - Amazon Athena – User Guide (AWS)
  - Athena FAQ (AWS)
- **Video**
  - Amazon Web Services: Data Analytics (LinkedIn)

Learn Python

Learning Python has been something I've postponed over the years. I've run scripts here and there and created small scripts but nothing serious. I'm dealing with a scenario where I need to expand my knowledge of it. I'm excited about the opportunity, so here we go!

This article assumes a working knowledge of the command line and at least one programming language.

The Basics

Python is an object-oriented, interpreted, and easy to learn language. The syntax is concise and approachable to beginners. I’ve used it in the past off and on and have found myself saying, “That’s it? Wow!”. You’ll need to install python to follow along.

Similar to Ruby, there is a pyenv utility for managing your python versions. It’s best to leave your system Python versions alone to avoid trouble with system dependencies.

```
#!/usr/bin/env python3
import pandas as pd

filename = "example.csv"
print(f"Chunking records for {filename}")
print("\n")
chunksize = 3
for chunk in pd.read_csv(filename, chunksize=chunksize):
    print(chunk)
```

Running this code requires a little more work. You’ll need to install pip so you can install the pandas library for data analysis tasks. I’m on macOS, so I need to run this command.

```
python3 -m pip install --user --upgrade pip
```

I need to install the pandas package.

```
pip3 install pandas
```

Download these files to run the code above and put them in the same directory.

example.csv & analyze-csv.py & .gitignore

Run this command in your terminal to make the program executable.

```
chmod +x ./analyze-csv.py
```

Run this command to list all your files.

```
ls -w
```

You should see output like this.

```
$ ls -w
analyze-csv.py          example.csv
```

Now execute the file.

```
./analyze-csv.py
```

You should see the following output.

```
$ ./analyze-csv.py
          store|sales
Office A            7
Office B            3
Office C            9
          store|sales
Office D          100
Office E            4
Office F           96
          store|sales
Office G           56
Office H           34
Office I           37
          store|sales
```

It didn't take much to create a program that can process CSV files in chunks.

Virtual Environments

I've learned that using a virtual environment is preferred over using the operating system's interpreter. The modern method to create a virtual environment is to use the venv package. Run this command in the directory you downloaded the **analyze-csv.py** file to. This will initialize your virtual environment.

```
python3 -m venv venv/
```

I ran into issues running this on a network drive. If the command hangs, try moving your folder to a physical disk.

Run this command to activate the environment.

```
source venv/bin/activate
```

You should see a shell prompt like this if it was successful. Note the (env) text indicating that the virtual environment is activated.

```
(venv) Jeffs-Laptop:python jeffbailey$
```

Now your program is isolated from the system. We can rest easy with the knowledge that our program will more reliably run in a virtual environment. Virtual environments save users of your program headaches. Users are not required to align their package versions with your required packages.

To leave the virtual environment run this command in your directory.

```
deactivate
```

Packages

Now that we have a virtual environment, we can install the pandas package but this time in the virtual environment.

```
pip install pandas
```

We can run this nifty command to create a requirements.txt file. This file tells users of our program what package versions to use.

```
pip freeze > requirements.txt
```

The freeze command will create a file that contains the dependencies you’ll need to run the **analyze-csv.py** file. Run this command to view the contents.

```
cat requirements.txt
```

You see an output like this.

```
$ cat requirements.txt
numpy==1.18.4
pandas==1.0.3
python-dateutil==2.8.1
pytz==2020.1
six==1.15.0
```

The pip freeze command lists all packages installed on your system. We’ve now defined which dependencies are required for the program to run. Users of your application can run this command to install the necessary modules.

```
 pip install -r requirements.txt
```

The virtual environment will keep us from having to use our system-wide installation of python, which may have *issues.*

Virtual environments will make our programs happy little clams.

Pushing It Up

If you like GitHub, you can create a GitHub repository called **learn-python** and push your new code to it. If you're in a hurry and already know all this stuff, you can clone my repository on GitHub and play around with it.

Run these commands to initialize a new repository and get it ready for the push.

```
git add . && git commit -am "Initial Commit"
```

Now let's push to our newly created repository. You'll want to replace my username with your GitHub username.

```
git remote add origin git@github.com:jeffabailey/learn-python.git
git push -u origin master
```

And that's that, you now have a program you can extend and have fun learning with.

Fundamentals

Learn Python without considering the fundamental programming constructs of Python? That’s crazy talk.

Variables

Variable assignment is simple enough, no magic here, just the way I like it.

The code is following the naming section of Google’s style guide. I am using code in the style guide source code as an example. Snake case variable naming is common when looking at Python code.

Comments

Comments for sanity. Tell your future self and your friends why you did something weird once upon a time.

Control Structures

We saw a for loop in our CSV analysis program. Here’s another example.

A while loop

An if else statement

Functions

Functions are the bread and butter of any programming language. I also threw in an f-string, which allows you to format your strings with variables elegantly.

Classes and Objects

Classes and objects are the nuts and bolts.

Lambdas

Lambdas smooth out the rough edges. Use them sparingly, most Python programmers are not a fan of them. Consider using a function rather than a lambda in more complex situations.

Exception Handling

Exception handling to keep from angering the user gods. Keep your errors under control and handle the flow.

Arrays

Arrays for life. Python arrays don’t contain any surprises, which makes them easy to use.

Operators

Operators for operating stuff.

Most Python operators are standard but watch out for type comparisons. Type comparisons with operators will compare the type names rather than the types themselves. Use the **isInstance** built in function to compare types instead.

There are many other constructs to learn in Python. It’s a joy to work with Python thanks to its natural and straightforward behavior.

Learn Python Beyond the Basics

There are thousands of Python books and loads of free tutorials online. Here are a couple of noteworthy titles that will help you learn effectively. I’ll leave you with The Zen of Python.

- **Books**
  - Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Python (O’Reilly)
  - Learning Python, 5th Edition (O’Reilly)
  - Python Cookbook, Third edition (O’Reilly)
  - Python Crash Course (Eric Matthes)
  - Head First Python: A Brain-Friendly Guide (Head First Books)
  - Learn Python 3 the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code (Zed Shaw’s Hard Way Series)
- **Videos**
  - Learning Python (LinkedIn)
  - Python Essential Training (LinkedIn)
  - Python Best Practices for Code Quality (Pluralsight)
  - Complete Python Developer in 2020: Zero to Mastery (Udemy)

Learning Elixir

Learning Elixir for fun and profit? It’s not a secret that Programmers like a great holy war. The echos of passionate programmers shouting with their banners high live on Stack

Overflow for eternity.

The editor wars rage on. Like many other holy wars, object-oriented vs. functional programming is an ongoing saga. Most programmers use functional programming even if they consider themselves object-oriented programmers. Many languages have become pragmatic and support both object-oriented and functional programming language paradigms.

I've enjoyed using the functional features of C# in the past and dabbled a bit with F#. I've never dove headfirst into a functional oriented language like Elixir. I mostly picked it up because I like The Pragmatic Programmer book and Dave Thomas as an author in general. I'm also interested in being able to do concurrent programming with less of a hassle.

Workflow

My workflow is a split window with the book and my IDE side by side on my Macbook. I'm trying things out as the material progresses and am running through the exercises.

My Workflow

Programming Elixir

As I traverse the Programming Elixir book, I feel like I'm learning regular expressions. These terse expressions are power-packed, look at this beauty.

```
# The long way
add_one = fn (n) -> n + 1 end
IO.puts add_one.(1)

# The short way
add_one = &(&1 + 1)
IO.puts add_one.(1)
```

Building quick little inline functions will take little effort. It’s a bit cryptic for my tastes, but if you’re doing quick and dirty code for a single task, this is a good thing.

Look at that, a few regular expressions and concise syntax, this code returns true.

```
match_end = & ~r/.*#{&1}$/
IO.puts "cat" =~ match_end.("t")
```

This code boggles my mind. I can have a set of regex libraries that are named appropriately. I can chain those expressions… what?!

I just convinced myself that learning Elixir is worth the investment.

Learning Elixir – Beyond the Basics

- **Books**
  - Programming Elixir by Dave Thomas
  - Elixir for Programmers by Dave Thomas

- **Visual Studio Code Extensions**
    - vscode-elixir (Syntax Highlighting, Intellisense)
    - Elixir Formatter (Code formatting)
    - Code Runner (Select and run code in your editor)