**The Shape We’re In**

Oct 29, 2017

**Intro**

We’ve all expended some good cycles and initiated the climb up the learning curve of PaaS development the past month or so. Let’s try to summarize our findings and figure out where we might go from here.

Vendors pitching the Cloud Development Stack for web deployments are many in number, but we’ve decided to focus our first round of discovery on

1. Amazon AWS/Elastic Beanstalk (native)
2. Microsoft Azure
3. IBM Bluemix
4. Heroku (based on the AWS PaaS)

We’ve also paid lip service to several others, including Google Apps, Oracle Cloud, but for now have concentrated on these.

Each of these PaaS platforms offers the neophyte developer (that would be us) an opportunity for little or no cost to test out their processes. As we envision developing something along the lines of the Compbobulator, the requirements include front end web design and presentation, database access, user dialog, and a robust object oriented programming language. Along with the PaaS choices, we are also examining the development frameworks that are also many in number; it is this cross matrix of PaaS plus framework that seems challenging. For now, we seem to have focused on some of the more popular frameworks, Python Flask and Python Django.

While this list looks focused and short, it is really an oversimplification of the challenge. Once the initial foray into the PaaS environment is made, a myriad of other decisions present themselves. Do we use virtual environments (optional but recommended by numerous “experts”) and containers, and adjunct environments like GitHub (source code library management and deployment) and what type of database world shall we choose (PostgreSQL for now). Do we understand the web architecture that imposes itself on the chosen combinations (not really, we apparently have much to learn) and how they help us and or hinder us. For each thread we pull, numerous additional threads appear, the initial welcoming smiling face turning quickly into Medusa.

**Observations and Questions Going Forward**

We are 3 enterprising, experienced and well rounded technology types. There appear to be a surprisingly large number of different paths to development and testing small scale web operations, perhaps as the scope of the task gets bigger, some choices become more obvious, and the path narrows. But at this point we have tested a few of the prominent PaaS platforms and the main thing we have learned is how much more we have to learn. In some ways, I find myself more confused than informed of what we have delved into.

Solving deployment issues for even the smallest apps appears to be quite challenging. It seems as though regardless of the “narrow stack” option chosen, a myriad of other technologies are used to accomplish the actual deployment and operation. For example, in deploying the Flask App Blueprint Pythia app on Bluemix, using Cloud Foundry as the deploy tool, the status and info messages during the actual deployment invoked Django and Heroku processes. I would guess this is CF doing its thing, for want of a better theory. Paul’s issues trying to solve a library issue on Azure indicate how cumbersome the deployment can be, invoking many moving parts and leaving an unsatisfying taste in one’s mouth when the path to correction escapes us.

One of the critical elements I see missing so far is where one does the “fancy” web front end, the user facing HTML insinuate itself into our app development? My experience with web development prior to our current adventure was using Visual Web Dev, where you designed the front end and hooked up the back end later. This seems the opposite and counter intuitive to me, all text editing and file arrangement and no visual aspect whatsoever. Yet all of our research and readings trumpet that these platforms and frameworks are the coolest, fastest, easiest way to develop web apps in today’s world.

Adding to the complexity of choosing specific frameworks and platforms, there seems to be a serious versioning compatibility challenge to developers like us. I’m old school, I remember when Fortran versions came out every 10 years, but were backward compatible for decades. These days, it seems the half life of any platform is 6 months to a year. What happens when Flask 1.13 stops talking with Postgres 9.1, and goes end of life with 4 months notice? Python 2.7 is EOL’ed shortly, how much work is out there porting 2.7 to 3.6?

We also seem to need more discovery along the lines of what is good practice versus what is required, vis a vis topics like containers and blueprints, the use of virtual environments, and the use of source code management (SCM) tools like Git/GitHub. Aside from an organizational tool, do they provide a functionality that boosts productivity and eliminates problems? Or are we merely substituting automated/less visible processes for manual processes in an effort to streamline our development?

Perhaps we have missed something in jumping right into deployment of simple apps. Should we have “schooled ourselves” more in the lingo and theory behind all this magic? Taken Web Apps for Dummies online and such? I certainly feel lost and confused a great deal of the time when trying to follow some of these allegedly beginner level tutorials.

Ego crushing as it is, do we continue plowing forward along these lines, looking for a next goal of a more complicated app deployment scenario, or do we venture outwards into different platforms to see if there is some commonality to the process across the various vendors and platforms?

Food for thought, and grist for the discussion mill.

**Addendum**

And is Oracle a potential new contender in the play yard? Meet the new boss, same as the old boss? Got this email this morning touting its prowess.

<https://blogs.oracle.com/developers/meet-the-new-application-development-stack-kubernetes-serverless-registry-cicd-java?elq_mid=91606&sh=91288241242322131522261115&cmid=WWMK170414P00004>

**Summary of Our Two Major Endeavors to Date**

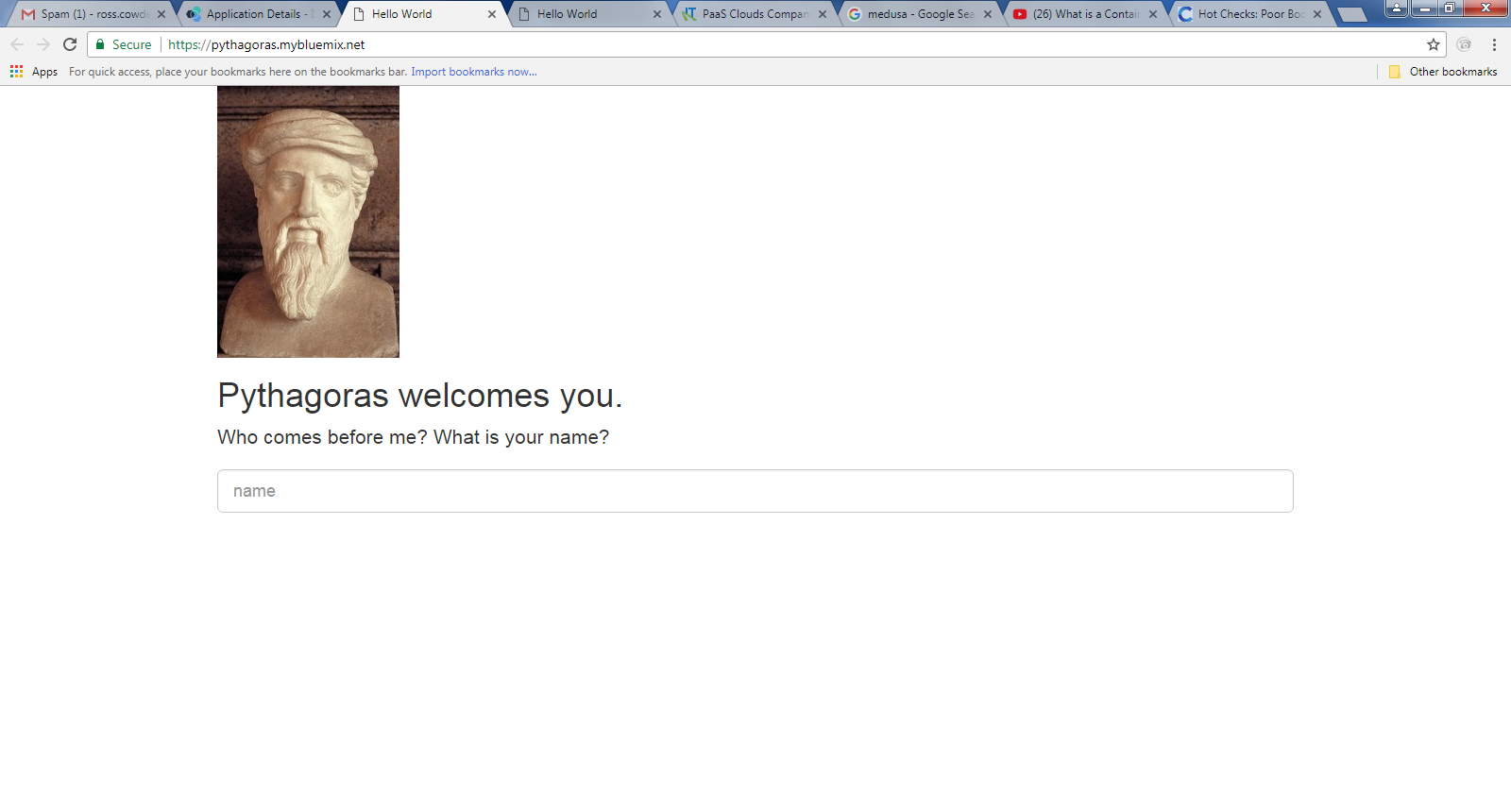
**Pass 1**

We resolved to sharpen our game by taking on one platform each and deploying a simple single web page that interrogated the user (front end) and took that nugget of user input to a backend database, interrogated it and returned a result to the screen. The plan was to develop, deploy and test the app locally, then move it on up to the Cloud. Sure sounded easy enough.

The essential steps to completion were

* Python project initiation and directory structure creation on local machine
* PostgreSQL db and user creation locally
* Python code manipulation to define user templates, manage db connection and data exchange
* Test, refine, repeat
* Deploy to the Cloud

Paul on BlueMix – deployed Pythagoras app, visually pleasing, accomplished all mission goals, relative effort and time to completion good, I would grade it A+



David on Azure – I believe David got his app going but can’t seem to find the emails describing it. David, feel free to insert your update here and grade yourself.

Ross On AWS – Got a basic database connection script to deploy and operate off of the page refresh cycle to communicate db output to the user. Used the same PostgreSQL db and code (developed and shared by Paul) as Pythagoras. No dialog per se, no visuals, no big deal. Grade C

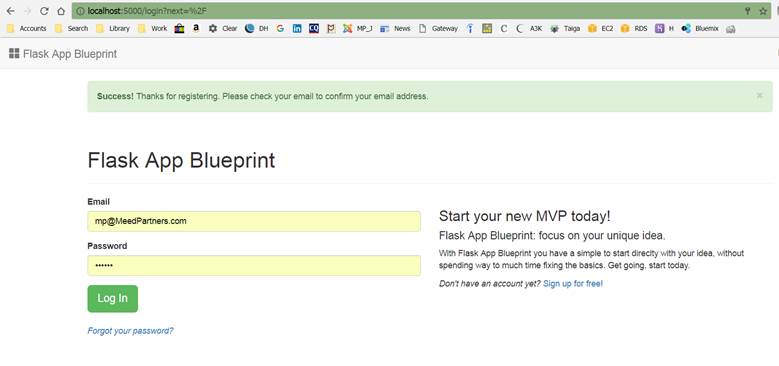
Overall, a good experience and some progress. PaaS platform setup for the projects was pretty easy, each offered a dashboard type front end that helps navigate the maze, and many of the PaaS features presented of are no current interest to our current discovery; e-commerce portals and such, along with arcane tunnels such as load balancing; for now, we are wishing for some load to balance.

Revealed was the incredible and baffling amount of technology combinations that present themselves to enterprising young men such as us. We learned that we have much, much more to learn.

**Pass 2**

Out of the kiddie pool, but not yet in the deep end (and it is quite deep), we developed a more focused approach: take a common app that Paul had developed locally, utilizing aspects of built in forms, Flask Blueprints, database interaction, a third party mailing plug in (SendMail), and building a user registration backend. The app, named Pythia, was to be deployed by the apparent nominal paradigm of local first, Cloud second. We decided to switch platforms from our original attempt as to develop some cross platform comparison perspective.

It should look something like this in operational form:

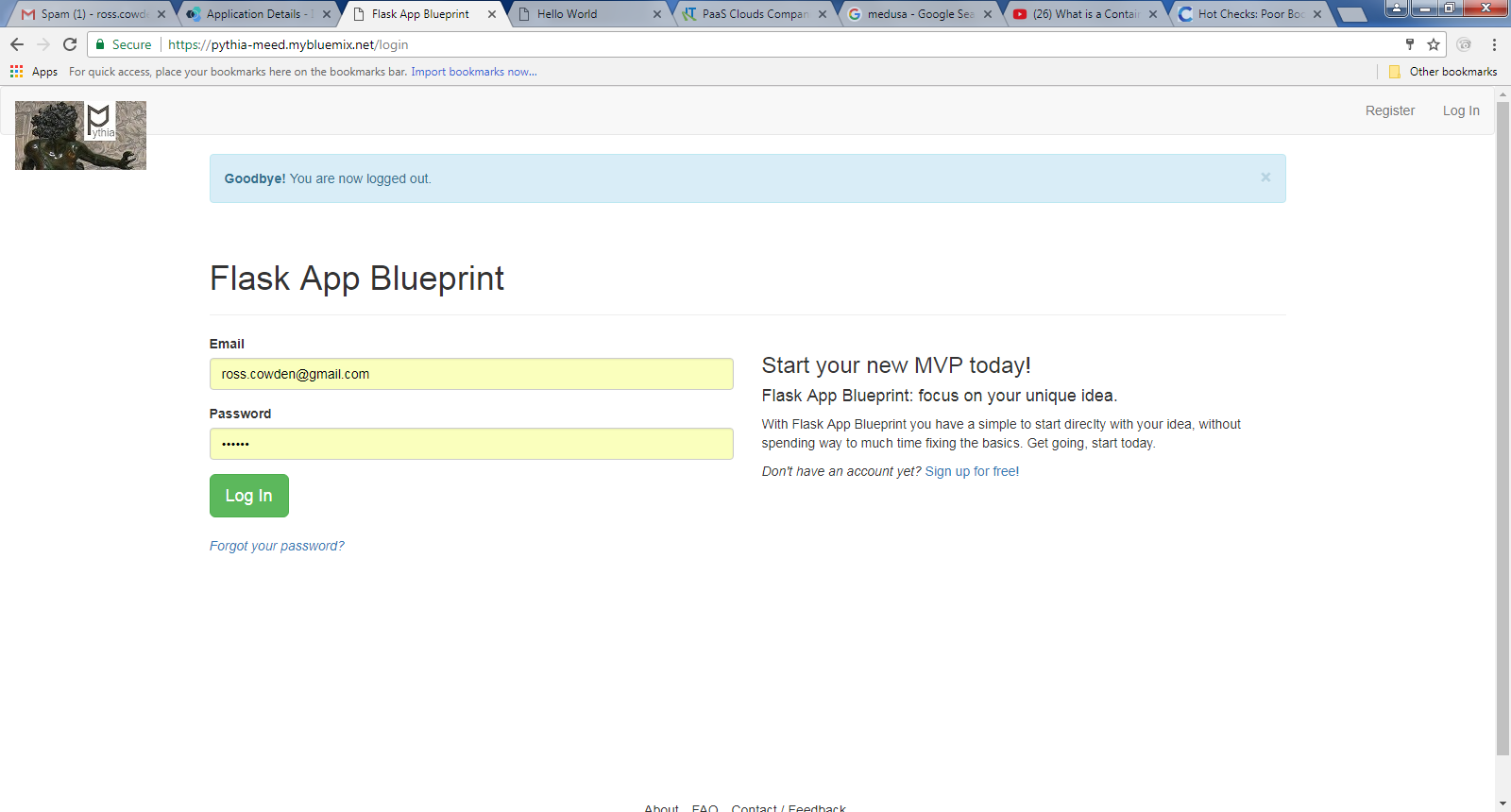


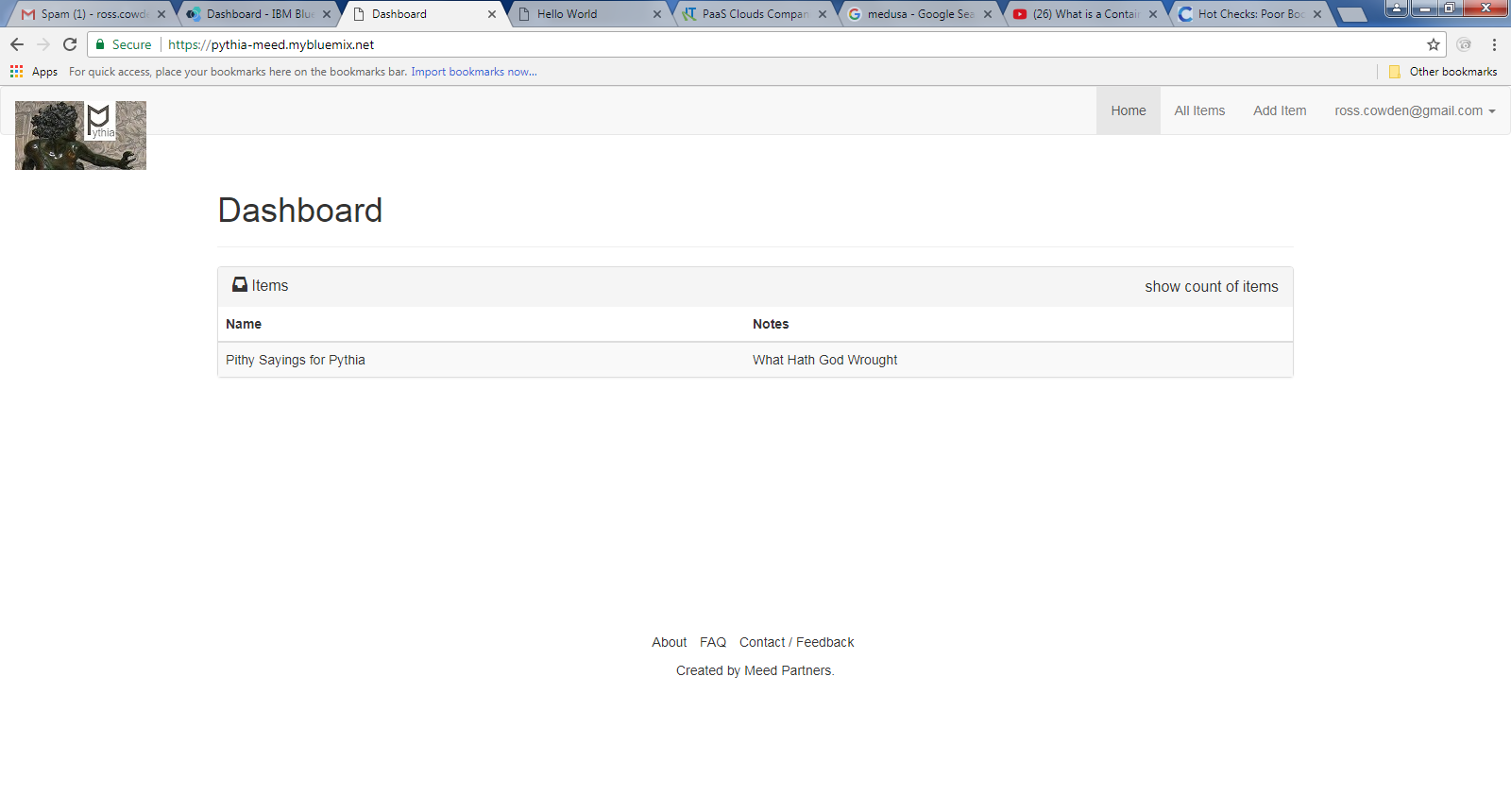
Paul on Azure – Seems to be hung up on some obscure and as yet unresolved library build issue. The irony is that it appears to be some kind of Microsoft C library problem and this is, of course, Mr. Softy’s platform. On hold for now, as Paul works for the Man. Ross to perhaps invest some cycles in this one as time permits this week.

David on AWS – Currently in deployment mode, investing some research cycles in Docker/container methodology, work to continue this week furthering this one.

Ross on Bluemix – Oddly enough, a quick and relatively painless deployment with the IBM platform. Key deployment observations:

Used the Cloud Foundry framework for the deploy, and found it to be a surprisingly pleasant experience. After establishing the environment pre-reqs, ferreting out the environment variable definitions, it was simply a cf push <appname> command from the local root directory of the app. Several unresolved/unknown libraries were encountered, fixed with the combo of pip install and pip freeze to produce a righteous requirements.txt file. Never did nail down a programmatic solution to either the environment variable definitions or database object creation, so they were done manually.





Then, boom, app is up and running. Grade A-. Bluemix/CloudFoundry gets the A, I get the minus