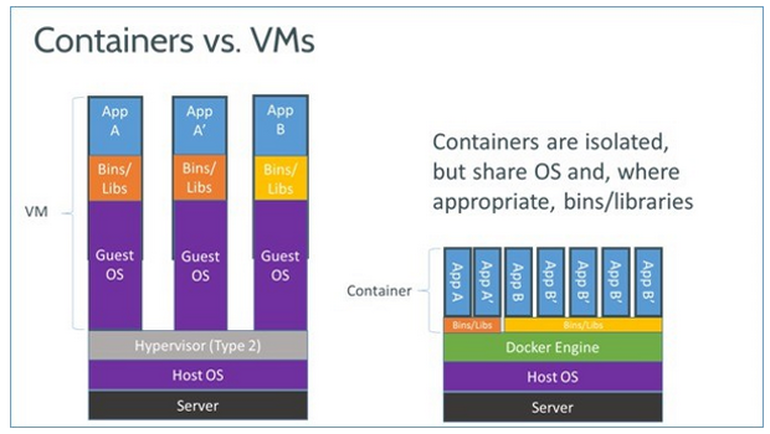
After facing many hurdles and creating a perfect machine learning model, most data scientists don’t realize that their real problem is in deploying that model onto production servers. Pushing machine learning model is nearly similar to pushing any other application to production, with a few stipulations. So, let’s take a look at one of the best options to deploy model on to production - using docker, Jenkins, and flask which are most commonly used in industries.

***Docker? But API is love, API is life!!!***

Hold your horses. We know that wrapping your model in flask and providing it as RESTful API service is more than enough for deployment purposes. But if there are any conflicting dependencies in your model libraries, then as soon as you deploy your model on production server, the model is likely to fail due to various reasons which are hard to find out.

If you’re an aspiring data scientist working with deep learning models, then you know how vexing it can get to install different versions of dependent libraries (like TensorFlow, Keras, and Pytorch) on your machine to train the model. For instance, TensorFlow runs on Python 2.7 only on linux distros but not on Windows. So, if you build the model on linux environment and deployed it on windows server, ***BOOM!*** All your efforts went in vain.

Since most deep learning libraries are still immature, there happen to have frequent updates. What’s bad is that some of your program may not work on the updated library. In order to keep current and updated version, Docker is a right answer. It containerizes the model with dependent libraries and environment, isolating it from underlying operating system and other applications. Unlike VMs, docker do not require any guest OS for each application, maintaining lightweight resource management.



Also, often when you are training a machine learning model, you build a model that can be trained with the right libraries and dependencies and model configurations on new incoming data continuously. As the data is streaming you keep track of the accuracies of the model on validation data from time to time and you might want to deploy new version model without reconfiguring the production servers. Again, docker comes to rescue as it plays a pivotal role in environment standardization - “*Build once, deploy everywhere*”.

***But, why Jenkins?***

******

Jenkins is a great open-source tool to deploy machine learning models. Written in Java, Jenkins was initially developed as a build and release tool. But the limitless potential of Jenkins made it the most popular CI/CD server.

Continuous Integration and Continuous Deployment (CI/CD) are best development practices followed by most industries today when they are developing applications. CI is the process of regularly commiting code changes into shared code repository and performing validation tests for each change before deploying it on production. CD is the process of automating this process and deploy the changes directly onto server if the code change passes all test cases.

So, If you need to fit the machine learning model in this CI/CD SDLC, Jenkins provides you all the tools to create jobs like automating validation testing that are triggered by events (like change in repository, updating model, etc.) and deploys it on production. It does all the work for you so you can focus on improving the model rather than designing test cases every time.

Jenkins also provides more than 1400 plugins including databases integration, Google cloud, Github, Bitbubket, etc. to support all kinds of tasks involving machine learning models.