**گام اول**

در قسمت اول، باید یک ایمیج بر پایه لینوکس را از داکرهاب بگیریم و روی آن امکان استفاده از دستور curl را ایجاد کنیم. برای این کار ابتدا باید ایمیج داکر ابونتو موردنظر خود را از داکرهاب بگیریم. برای این کار از دستور pull استفاده میکنیم:

docker pull ubuntu:focal

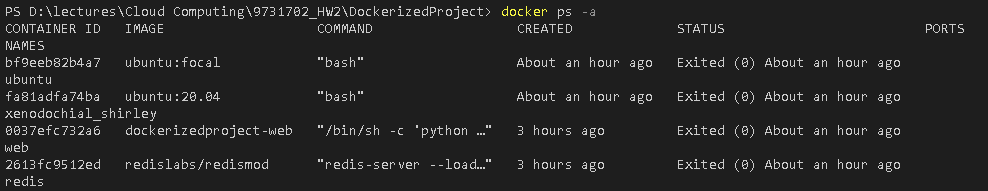
وقتی ایمیج مورد نظر گرفته شد، با دستور زیر آن را اجرا میکنیم:

docker run -it --name ubuntu ubuntu:focal bash

در دستور بالا با استفاده از -it وقتی ابونتو بالا می آید، محیط اجرایی و امکان اجرای دستورات در ترمینال آن را به ما میدهد درحالیکه اگر فقط از دستور docker run ubuntu:focal استفاده کنیم، چون خود لینوکس کاری برای اجرا ندارد، کانتینر آن بالا می آید و اجرا شده و تمام میشود و امکان استفاده از ترمینال آن را به ما نمیدهد. در ترمینال ایمیج لینوکس برای ایجاد امکان استفاده از دستور curl از دستور زیر استفاده میکنیم:

apt-get update && apt-get install curl

apt-get پکیج منجر لینوکس است که اول پکیج های ایمیج را آپدیت میکنیم و بعد curl را نصب میکنیم. اگر در این حالت از ایمیج exit کنیم، دستور docker ps خالی است چون ایمیجی در حال اجرا نیست اما اگر دستور docker ps -a را بزنیم، خروجی به صورت زیر است:

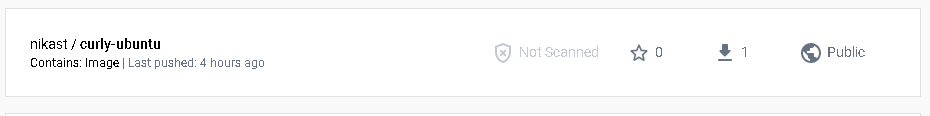


که نشان داده که ایمیج ابونتو یک ساعت پیش ساخته شده است. حال برای اینکه این ایمیج را روی داکرهاب قرار دهیم، ابتدا به آن یک تگ میدهیم. برای این کار از دستور زیر استفاده میکنیم:

docker tag curly-ubuntu:1.0 nikast/curly-ubuntu:1.0

سپس آن را با دستور زیر روی داکرهاب پوش میکنیم:

docker image push nikast/curly-ubuntu:1.0



اکنون این ایمیج را با دستور زیر از داکرهاب گرفته و آن را اجرا میکنیم:

docker run -it nikast/curly-ubuntu:1.0 bash

اکنون دستور curl google.com را اگر بزنیم، خروجی زیر را میگیریم:

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">

<TITLE>301 Moved</TITLE></HEAD><BODY>

<H1>301 Moved</H1>

The document has moved

<A HREF="http://www.google.com/">here</A>.

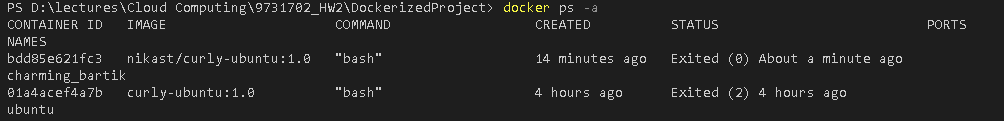
</BODY></HTML>

و اگر دستور curl [www.google.com](http://www.google.com/) را بزنیم، خروجی زیر را میگیریم:



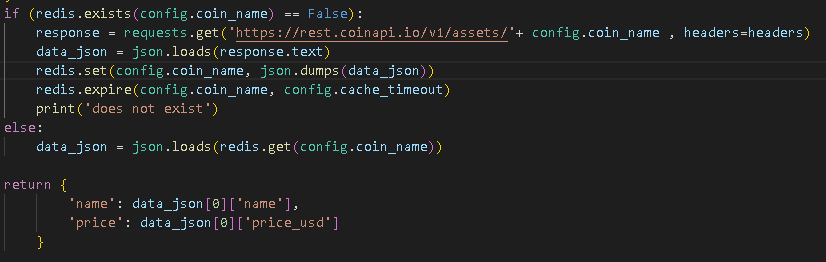
Figure 1- بخشی از خروجی curl www.google.com

لیست ایمیج های موجود در سیستم خود به صورت زیر است:

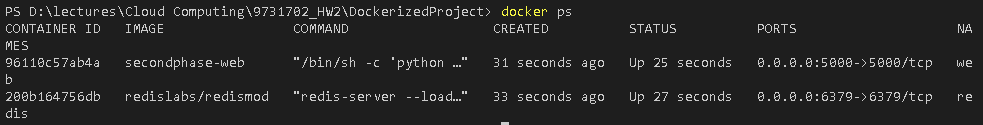


**گام دوم**

در گام دوم باید سروری برای دریافت قیمت ارز های دیجیتال می ساختیم. در این بخش باید از قابلیت ردیس برای کش کردن اطلاعات استفاده میکردیم. برای این کار، در فایل config این زمان انقضا تنظیم میشود و اگر ارز موردنظر قبلا درخواست برای آن ارسال نشده بود، در ردیس ذخیره میشود و زمان انقضای آن از آن لحظه شروع میشود. اگر این ارز قبلا درخواست داده شده بود، دیگر به api سایت coinapi کال زده نمیشود و اطلاعات ارز موردنظر از ردیس خوانده و نمایش داده میشود.



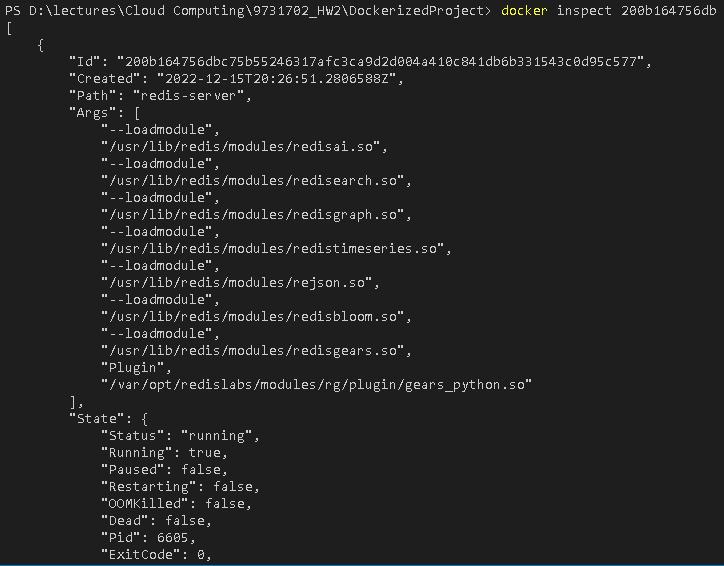
دستور docker ps برای نمایش containerهای موجود به صورت زیر است:



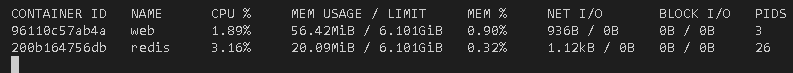
همانطور که مشاهده میشود، برنامه دارای دو کانتینر وب و ردیس می‌باشد. همجنین دستور docker inspect برای ایمیج سرور و کانتینر وب به صورت زیر است:



و برای کانتینر ردیس به صورت زیر می باشد:



خروجی دستور docker stats برای نمایش منابع در حال استفاده به صورت زیر می باشد:



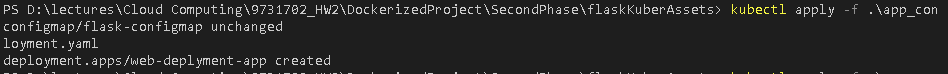
آدرس زیر نیز برای persist کردن اطلاعات به عنوان volume استفاده میشود:



**گام سوم**

در گام سوم برای هر یک از سرویس های وب و ردیس فایل های مربوطه را ساخته و با applyکردن آن ها در kubectl کانتینتر مربوط به آنها را در minikube ایجاد میکنیم. نمونه ای از apply شدن فایل ها در تصویر زیر نشان داده شده است.

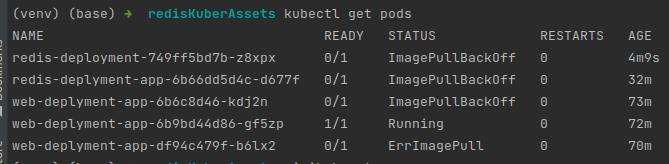


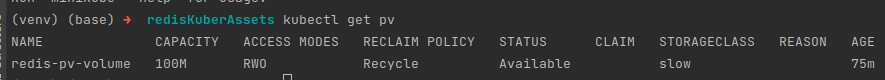


برای مشاهده پادهای موجود دستور

kubectl get pods

را میزنیم که خروجی آن به صورت زیر است:

با دستور kubectl get pv والیوم های موجود را چک میکنیم:

با دستور kubectl get pvc داریم: