گزارش پنجم

مقدمه

به‌طورکلی داکر[[1]](#footnote-1) یک سکو برای توسعه، انتقال و اجرای برنامه‌ها است. با کمک داکر می­توانیم نرم­افزار­ها را مستقل از زیرساخت به‌سرعت پیاده‌سازی کنیم. همچنین با کمک این نرم‌افزار می­توان زیرساخت و سخت‌افزار مورداستفاده در پروژه‌ها را مدیریت کرد و فاصله زمانی میان پیاده‌سازی و رسیدن به مرحله محصول نهایی را به حداقل رساند.

داکر توانایی بسته‌بندی و اجرای برنامه­های ما را در محیط ایزوله‌ای به نام کانتینر[[2]](#footnote-2) فراهم می­کند. این ایزوله شدن شرایطی ایجاد می­کند که کانتینرها را بتوان به‌صورت هم‌زمان روی کامپیوتر میزبان اجرا کرد. این کانتینرها در مقابل ماشین‌های مجازی بسیار سبک هستند و می­توانند بدون نظارت بر روی کامپیوتر میزبان اجرا شوند. به‌طورکلی داکر ابزار و سکویی برای مدیریت کردن این کانتینرها است تا توسعه سریع و ساده­تر شود و بتوان پردازش قابل توسعه­ای، روی خوشه‌ای از کامپیوترها اجرا کرد. همچنین هنگامی‌که برنامه کامل شد و به مرحله محصول نهایی رسید، بدون در نظر گرفتن نوع زیرساخت، قابل‌انتقال به محیط‌های دیگر (مرکز داده‌ها، شرکت‌های خدمات دهنده ابری،...) باشد.

به‌طورکلی هسته اصلی داکر از سه بخش تشکیل‌شده که در شکل ‏4–5 قابل‌مشاهده است، بخش اول یک سرویس‌دهنده است که مدیریت اشیا ساخته‌شده توسط داکر را بر عهده دارد، اشیائی همچون فایل ایمیج (فشرده‌شده کانتینرها)، کانتینرها، شبکه‌ها و فضاهایی که تخصیص داده می­شوند. این برنامه فرایند محاسباتی[[3]](#footnote-3) نام‌گذاری می­شود. بخش دوم یک برنامه رابط است (REST API) که دستوراتی که ما می­دهیم را به فرایند محاسباتی منتقل می­کند. بخش سوم یک رابط و خط دستور (CLI)[[4]](#footnote-4) است که فرمان‌های ما در آن وارد می­شود. خط دستور از یک REST API استفاده می­کند تا وظیفه کنترل و ارتباط با فرایند محاسباتی را یا با استفاده از فرمان­هایی از پیش نوشته‌شده یا به‌صورت واردکردن در خط دستور انجام دهد.



شکل ‏4–5-موتور و هسته داکر که روند فرایند مدیریت برنامه را مشخص می­کند. [22]

به سه دلیل اصلی در این پروژه برای پیاده‌سازی فرایند دریافت و ذخیره‌سازی اطلاعات از داکر استفاده شد که به شرح زیر است:

پیشرفت و توسعه مداوم برنامه‌ها؛

داکر با کار در محیط‌های استاندارد و با استفاده از کانتینرهای محلی این اجازه را به توسعه‌دهندگان برای پیاده­سازی برنامه‌های پیچیده می­دهد و چرخه عمر توسعه را ساده می­کند. چراکه بخش‌های مختلف برنامه می­توانند مستقل از هم با نسخه‌ها و حتی زبان­های برنامه‌نویسی متفاوت پیاده­سازی و اجرا شوند. به این عمل فرایند یکپارچه‌سازی پیوسته[[5]](#footnote-5)/ تحویل پیوسته[[6]](#footnote-6) می­گویند.

بستر مبتنی بر کانتینرهای داکر، ایجاد برنامه‌هایی قابل‌حمل[[7]](#footnote-7) را امکان‌پذیر می‌کند. کانتینرهای داکر می‌توانند بر روی کامپیوتر محلی توسعه‌دهنده، در ماشین‌های فیزیکی یا مجازی در یک مرکز داده، بر روی ارائه‌دهندگان خدمات ابری یا در مخلوطی از این محیط‌ها اجرا شوند که کمک می­کند برنامه به‌سادگی روی کامپیوتر توسعه‌دهنده آماده و سپس برای پیاده‌سازی عملی به سرورهایی باقدرت و پاسخ‌گویی بیشتر منتقل شود.

کانتینرهای داکر سبک و سریع هستند چراکه تنها برنامه‌هایی که موردنیاز است را در خود جای‌داده‌اند. این‌یک جایگزین مناسب و مقرون‌به‌صرفه به‌جای ماشین‌های مجازی است که نیازمند پیش‌نیازهای بیشتری برای پیاده‌سازی محیط‌های ایزوله هستند؛ بنابراین می­توان از ظرفیت محاسبه بیشتری برای رسیدن به اهداف خود استفاده کرد. داکر برای برنامه‌هایی با چگالی بالا (بخش‌های متفاوت) و عملکردهای کوچک و متوسط که نیاز به انجام کارهای بیشتر با منابع کمتری دارند، بسیار مناسب است.

داکر از یک معماری کاربر-سرویس­دهنده­ای استفاده می­کند. به این شکل که کاربر داکر، با فرایند محاسباتی در ارتباط است که وظیفه ساخت و اجرا و توزیع کانتینرها را بر عهده دارد. کاربر داکر و فرایند اصلی می­توانند در یک کامپیوتر باشند یا می­توانند از طریق REST API یا رابط­های شبکه با یکدیگر در ارتباط باشند. به‌طورکلی برنامه‌ها در محیطی به نام کانتینر اجرا می­شوند که نمونه‌ای قابل‌اجرا از فایلی به نام ایمیج است. برای ساخت فایل ایمیج یا باید خودمان آن فایل را از ابتدا ایجاد کنیم، یا از ایمیج های استانداری که در بخش رجیستری داکر (داکرهاب) قرارگرفته استفاده کنیم. عموماً برای سبک بودن محیط اجرای کار حتی درصورتی‌که بخواهیم فایل ایمیجی را خودمان طراحی کنیم از یک ایمیج مبتنی بر لینوکس ساده شروع کرده و هر چیزی که به آن نیاز داریم را به آن اضافه می­کنیم، ولی در اکثر مواقع برنامه‌های موردنیاز در داکرهاب وجود دارند که می­توان از آن بهره­برد.

برای مثال در شکل ‏4–6 با دستور اول یک فایل ایمیج ساخته می­شود. با دستور دوم یک فایل ایمیج از بخش رجیستری (داکرهاب) به کامپیوتر میزبان منتقل می­شود و با دستور سوم، یکی از فایل­های ایمیج ساخته‌شده به‌صورت کانتینر درآمده و اجرا می­شود.

به این شکل کانتینرها ایجاد می­شوند. حال برای ارتباط کانتینرها با یکدیگر از مکانیزم های دیگری که پیش­تر در شکل ‏4–5 دیدیم استفاده می­شود. مکانیزم هایی همچون ایجاد شبکه یا اجازه دسترسی به حجم‌های فیزیکی که ارتباط میان کانتینرها را فراهم کند.

از پیش­تر در بخش مروری بر داکر به یاد داریم که می­توان ایمیج های استاندارد را از داکرهاب، بارگذاری و مورداستفاده قرارداد. RabbitMQ خود یکی از ایمیج های استاندارد است که به‌سادگی از آن استفاده می­شود. برای این کار می­توانیم در داخل داکر فایل از ایمیج استاندارد شروع کرده و پلاگین­هایی که قصد فعال‌سازی آن را داریم، با اجرای دستور مربوطه به‌کارگیریم تا ایمیج جدیدی با ویژگی‌هایی که مدنظر داریم ساخته شود. در شکل زیر محتوای داکر فایل را می‌توان مشاهده کرد.

FROM rabbitmq:3-management

RUN rabbitmq-plugins enable rabbitmq\_management

RUN rabbitmq-plugins enable  rabbitmq\_mqtt –offline

محتوای داکر فایل برای راه‌اندازی RabbitMQ

حال با به­کارگیری داکر کامپوز تنها کافی است دستور دهیم که ایمیج فوق ساخته شود و پورت مخصوص به ارتباط با کانتینر و کامپیوتر میزبان برقرار شود. همچنین می­توان فایل تنظیمات را تغییر داد و آن را در داخل کانتینر قرار داد. در شکل ‏4–15 می­توان دستوراتی که در فایل docker-compose.yaml برای بالا آمدن کانتینر RabbitMQ نوشته‌شده را مشاهده کرد.

  queue:

    build: rbmq/.

    container\_name: rabbitmq

    ports:

      - 5672:5672

      - 15672:15672

      - 1884:1884

    volumes:

      - "./rbmq/Conf/:/etc/rabbitmq/"

بخش مرتبط با RabbitMQ در داکر کامپوز

1. Docker [↑](#footnote-ref-1)
2. container [↑](#footnote-ref-2)
3. Daemon process [↑](#footnote-ref-3)
4. Command line [↑](#footnote-ref-4)
5. CI(continuous integration) [↑](#footnote-ref-5)
6. CD(continuous delivery) [↑](#footnote-ref-6)
7. Portable [↑](#footnote-ref-7)