Docker: مفاهیم و تکنیک‌های پیشرفته در Docker Compose

مجموعه نکته‌های رسمی و علمی

تهیه شده توسط Express24.ir

مقدمه‌ای بر داکر و کانتینرسازی

1. داکر به عنوان یک بستر متن‌باز، امکان توسعه، اجرا و مدیریت کاربردها را در محیط‌های ایزوله و قابل حمل موسوم به کانتینر فراهم می‌آورد.

2. مفهوم کانتینرسازی بر اساس مجازی‌سازی در سطح سیستم‌عامل بنا شده است، که در آن منابع مشترک کرنل میزبان میان کانتینرهای مجزا تقسیم می‌گردند.

3. تفاوت اساسی کانتینرها با ماشین‌های مجازی در عدم نیاز به یک سیستم‌عامل کامل برای هر نمونه است، که منجر به سبکی و سرعت بالای راه‌اندازی می‌گردد.

4. یکی از مزایای کلیدی داکر، تضمین یکپارچگی محیط اجرایی از توسعه تا استقرار است، که مشکلات "در سیستم من کار می‌کند" را به حداقل می‌رساند.

5. کارایی منابع در کانتینرها به دلیل اشتراک کرنل و عدم سربار اضافی سیستم‌عامل، به مراتب بالاتر از ماشین‌های مجازی سنتی است.

6. اجزای اصلی داکر شامل Dockerfile برای تعریف ساختار ایمیج، ایمیج‌ها به عنوان قالب‌های قابل اجرا، و کانتینرها به عنوان نمونه‌های زنده ایمیج‌ها می‌باشند.

7. معماری داکر از سه بخش اصلی Docker Engine (شامل CLI، Daemon و API)، Docker Images و Docker Registries (مانند Docker Hub) تشکیل شده است.

8. ایزوله‌سازی فرآیندها و وابستگی‌ها در کانتینرها، امنیت و پایداری کاربردها را افزایش داده و از تداخل‌های ناخواسته جلوگیری می‌نماید.

9. داکر هاب (Docker Hub) به عنوان یک رجیستری مرکزی، امکان به اشتراک‌گذاری عمومی و خصوصی ایمیج‌های داکر را برای جامعه کاربران فراهم می‌آورد.

10. استفاده از داکر به طور فزاینده‌ای در معماری میکروسرویس‌ها و بسترهای ارکستراسیون کانتینر مانند کوبرنتیس برای تسهیل مقیاس‌پذیری و مدیریت کاربردها رایج است.

چرا از داکر استفاده کنیم؟

1. استفاده از داکر، تسهیل فرآیند استقرار و مقیاس‌پذیری برنامه‌ها را از طریق کانتینری‌سازی فراهم می‌آورد.

2. داکر امکان ایجاد محیط‌های توسعه یکپارچه و قابل تکرار را برای توسعه‌دهندگان فراهم می‌کند، که منجر به کاهش مشکلات ناشی از تفاوت‌های محیطی می‌شود.

3. کانتینرهای داکر، به دلیل ایزوله بودن، امنیت بیشتری نسبت به روش‌های سنتی مجازی‌سازی ارائه می‌دهند و از تداخل بین برنامه‌ها جلوگیری می‌کنند.

4. داکر، با استفاده از سیستم لایه‌ای تصاویر، بهینه‌سازی مصرف منابع سیستم را ممکن ساخته و حجم تصاویر را به حداقل می‌رساند.

5. فرآیند مدیریت نسخه‌ها در داکر به واسطه تصاویر و رجیستری‌ها به صورت متمرکز و کارآمد انجام می‌پذیرد.

6. داکر، امکان انتقال سریع و آسان برنامه‌ها بین محیط‌های مختلف (توسعه، تست، و تولید) را فراهم می‌سازد، که باعث تسریع چرخه توسعه می‌شود.

7. استفاده از داکر، هزینه‌های زیرساختی را به دلیل استفاده بهینه از منابع سخت‌افزاری و کاهش نیاز به سرورهای متعدد، کاهش می‌دهد.

8. داکر، با فراهم کردن ابزارهای مدیریت کانتینر، امکان نظارت و کنترل دقیق بر عملکرد برنامه‌ها را میسر می‌سازد.

9. داکر، با پشتیبانی از استانداردهای صنعتی، امکان یکپارچگی با سایر ابزارها و فناوری‌های توسعه و استقرار نرم‌افزار را فراهم می‌کند.

10. استفاده از داکر، امکان ایجاد معماری‌های میکروسرویس را تسهیل نموده و به توسعه‌دهندگان اجازه می‌دهد تا برنامه‌های بزرگ را به اجزای کوچک‌تر و قابل مدیریت تقسیم کنند.

مفاهیم اصلی کانتینر

1. کانتینرسازی، با تمرکز بر مفهوم Docker، به عنوان راهکاری نوین در توسعه و استقرار نرم‌افزار، امکان بسته‌بندی مستقل برنامه‌ها و وابستگی‌هایشان را در محیط‌های ایزوله فراهم می‌آورد.

2. یکی از اصول بنیادین کانتینرها، جداسازی و ایزوله‌سازی فرآیندها و منابع سیستم‌عامل است که این امر از طریق لایه‌بندی و استفاده از ویژگی‌های هسته سیستم‌عامل صورت می‌پذیرد.

3. تصویر (Image) در Docker، قالبی خوانا و تغییرناپذیر است که شامل مجموعه‌ای از دستورالعمل‌ها برای ایجاد یک کانتینر می‌باشد و لایه‌های متعددی را در بر می‌گیرد.

4. کانتینر (Container)، نمونه‌ای قابل اجرا از یک ایمیج است که شامل تمامی اجزای مورد نیاز برای اجرای یک برنامه، از جمله کد، زمان اجرا، کتابخانه‌ها و ابزارهای سیستمی است.

5. مدیریت چرخه حیات کانتینرها، شامل ایجاد، اجرا، توقف، حذف و مانیتورینگ، توسط Docker Engine انجام می‌شود که هسته اصلی فناوری Docker محسوب می‌گردد.

6. شبکه‌بندی در Docker به کانتینرها اجازه می‌دهد تا با یکدیگر و با دنیای خارج ارتباط برقرار کنند، که این قابلیت از طریق مکانیزم‌های مختلف شبکه‌سازی پیشرفته امکان‌پذیر است.

7. ذخیره‌سازی پایدار (Persistent Storage) برای کانتینرها از طریق Volumeها و Bind Mounts مدیریت می‌شود تا از اتلاف داده‌ها در صورت توقف یا حذف کانتینر جلوگیری شود.

8. Docker Compose ابزاری برای تعریف و اجرای برنامه‌های چندکانتینری است که به کاربران امکان می‌دهد پیکربندی و وابستگی‌های سرویس‌ها را در قالب یک فایل YAML تعریف نمایند.

9. مفهوم ارکستراسیون کانتینر، که ابزارهایی مانند Kubernetes پیشتاز آن هستند، مدیریت، مقیاس‌پذیری و خودکارسازی استقرار تعداد زیادی کانتینر را در محیط‌های تولیدی تسهیل می‌کند.

10. استفاده از Docker منجر به افزایش چشمگیر قابلیت حمل (Portability)، سازگاری محیطی (Environment Consistency) و سرعت در فرآیندهای توسعه و استقرار نرم‌افزار می‌گردد.

مفاهیم اصلی ایمیج (Image)

1. ایمیج در داکر، یک بستهٔ ایستا و قابل حمل است که شامل تمامی مؤلفه‌ها و وابستگی‌های لازم برای اجرای یک برنامهٔ نرم‌افزاری می‌شود.

2. ایمیج‌ها از لایه‌های مختلفی تشکیل شده‌اند که هر لایه نشان‌دهندهٔ تغییرات ایجاد شده در فایل سیستم است و با استفاده از دستورالعمل‌های Dockerfile ساخته می‌شوند.

3. ایجاد ایمیج‌ها بر اساس یک فایل Dockerfile انجام می‌شود که حاوی دستورالعمل‌هایی برای نصب نرم‌افزار، کپی فایل‌ها، تنظیمات محیطی و اجرای دستورات است.

4. هر ایمیج دارای یک شناسه (ID) منحصر به فرد است که برای شناسایی و مدیریت آن در سامانه داکر استفاده می‌شود.

5. ایمیج‌ها می‌توانند از طریق رجیستری‌های عمومی یا خصوصی داکر به اشتراک گذاشته شوند و در دسترس سایر کاربران قرار گیرند.

6. ایمیج‌ها در حالت پیش‌فرض، immutable (تغییرناپذیر) هستند و تغییرات در آنها تنها با ایجاد یک ایمیج جدید امکان‌پذیر است.

7. ایمیج‌ها امکان ایجاد محیط‌های ایزوله و یکنواخت برای اجرای برنامه‌ها را فراهم می‌کنند، که این امر منجر به افزایش قابلیت حمل و سازگاری برنامه‌ها می‌شود.

8. ایمیج‌ها با استفاده از لایه‌های مشترک، فضای ذخیره‌سازی را بهینه می‌کنند و امکان استفاده مجدد از داده‌ها را فراهم می‌آورند.

9. ساخت ایمیج‌ها به صورت افزایشی انجام می‌شود، بدین معنی که تنها لایه‌هایی که تغییر کرده‌اند، دوباره ساخته می‌شوند و این فرآیند باعث افزایش سرعت ساخت ایمیج می‌شود.

10. ایمیج‌ها می‌توانند بر اساس ایمیج‌های پایه (base images) ساخته شوند، که این امر فرآیند ساخت ایمیج‌ها را ساده‌تر و سریع‌تر می‌کند.

مفاهیم اصلی ریپازیتوری (Registry)

1. رجیستری داکر (Docker Registry) به عنوان یک سامانه متمرکز برای ذخیره‌سازی، توزیع و مدیریت ایمیج‌های داکر عمل می‌کند که امکان دسترسی و اشتراک‌گذاری ایمیج‌ها را در محیط‌های مختلف فراهم می‌آورد.

2. یکی از کاربردهای اساسی رجیستری، تضمین یکپارچگی و امنیت ایمیج‌ها از طریق مکانیزم‌های احراز هویت و کنترل دسترسی است که از دسترسی غیرمجاز به منابع جلوگیری می‌کند.

3. تمایز بین "رجیستری" و "مخزن" (Repository) حائز اهمیت است؛ رجیستری شامل مجموعه‌ای از مخازن است، در حالی که هر مخزن به یک ایمیج خاص و نسخه‌های مختلف آن (تگ‌ها) اختصاص دارد.

4. درحالی‌که رجیستری‌های عمومی نظیر Docker Hub امکان اشتراک‌گذاری جهانی ایمیج‌ها را فراهم می‌آورند، رجیستری‌های خصوصی برای مدیریت ایمیج‌های سازمانی و حفظ محرمانگی طراحی شده‌اند.

5. نسخه‌بندی ایمیج‌ها در رجیستری از طریق تگ‌گذاری (Tagging) صورت می‌گیرد، که امکان ارجاع دقیق به نسخه‌های مختلف یک ایمیج و ردیابی تغییرات آن را میسر می‌سازد.

6. مدیریت چرخه حیات ایمیج‌ها، شامل بارگذاری (push)، دریافت (pull)، و حذف (delete)، از قابلیت‌های کلیدی رجیستری است که به سازماندهی مؤثر منابع کمک می‌کند.

7. پشتیبانی از پروتکل HTTPS در رجیستری، امنیت ارتباطات بین کلاینت داکر و سرور رجیستری را تضمین می‌کند و از شنود یا دستکاری داده‌ها جلوگیری به عمل می‌آورد.

8. مفهوم "لایه‌های ایمیج" (Image Layers) به طور مستقیم با عملکرد رجیستری مرتبط است، زیرا رجیستری بهینه‌سازی ذخیره‌سازی را با نگهداری تنها یک کپی از لایه‌های مشترک انجام می‌دهد.

9. پیکربندی رجیستری‌های خصوصی امکان ادغام با سیستم‌های مدیریت هویت و دسترسی موجود در سازمان را فراهم می‌کند که به یکپارچگی زیرساخت امنیتی کمک می‌نماید.

10. قابلیت "وب‌هوک" (Webhook) در برخی رجیستری‌ها، امکان اعلان خودکار تغییرات در مخازن را به سیستم‌های دیگر (مانند سیستم‌های CI/CD) فراهم آورده و فرایندهای توسعه را تسریع می‌بخشد.

نصب داکر روی سیستم عامل‌ها

1. پیش‌نیازهای سیستمی جهت نصب موتور داکر (Docker Engine) در توزیع‌های لینوکس، شامل معماری ۶۴ بیتی پردازنده و نسخه کرنل ۳.۱۰ یا بالاتر جهت پشتیبانی از قابلیت‌های کلیدی چون فضاهای نام (namespaces) و گروه‌های کنترلی (cgroups) می‌باشد.

2. در سیستم‌عامل‌های ویندوز، نصب Docker Desktop مستلزم فعال‌سازی فناوری مجازی‌سازی سخت‌افزاری (Hardware Virtualization) در BIOS/UEFI و بهره‌گیری از زیرسیستم ویندوز برای لینوکس نسخه ۲ (WSL 2) یا Hyper-V به عنوان پس‌زمینه (backend) اجرایی است.

3. فرآیند نصب داکر در توزیع‌های مختلف لینوکس به دلیل تفاوت در مدیران بسته (package managers) نظیر APT برای سیستم‌های مبتنی بر دبیان و YUM/DNF برای سیستم‌های مبتنی بر RHEL، نیازمند پیروی از دستورالعمل‌های اختصاصی هر توزیع است.

4. پس از اتمام نصب، افزودن کاربر غیر ریشه (non-root user) به گروه کاربری `docker` یک اقدام امنیتی و عملیاتی استاندارد محسوب می‌شود که نیاز به استفاده از دستور `sudo` برای تعامل با داکر دیمن (Docker daemon) را مرتفع می‌سازد.

5. اعتبارسنجی نصب موفقیت‌آمیز و عملکرد صحیح داکر، به طور متداول از طریق اجرای ایمیج آزمایشی `hello-world` انجام می‌شود که صحت ارتباط کلاینت با دیمن و قابلیت واکشی (pull) ایمیج از رجیستری عمومی را تأیید می‌کند.

6. در محیط‌های تولیدی (production)، توصیه اکید بر استفاده از مخازن رسمی (official repositories) داکر به جای اسکریپت‌های نصب سریع (convenience scripts) است تا مدیریت نسخه‌ها و اعمال به‌روزرسانی‌های امنیتی به صورت کنترل‌شده انجام پذیرد.

7. برای سیستم‌عامل macOS، ابزار Docker Desktop از هایپروایزر بومی HyperKit که بر پایه Hypervisor.framework اپل توسعه یافته، برای ایجاد یک ماشین مجازی لینوکسی سبک‌وزن و اجرای داکر دیمن در آن بهره می‌برد.

8. پیکربندی پیشرفته داکر دیمن پس از نصب، از طریق فایل `daemon.json` واقع در مسیر `/etc/docker/` صورت می‌پذیرد و امکان تنظیم پارامترهایی چون درایور ذخیره‌سازی (storage driver)، پل‌های شبکه (network bridges) و آدرس‌های رجیستری خصوصی را فراهم می‌کند.