



جاودان

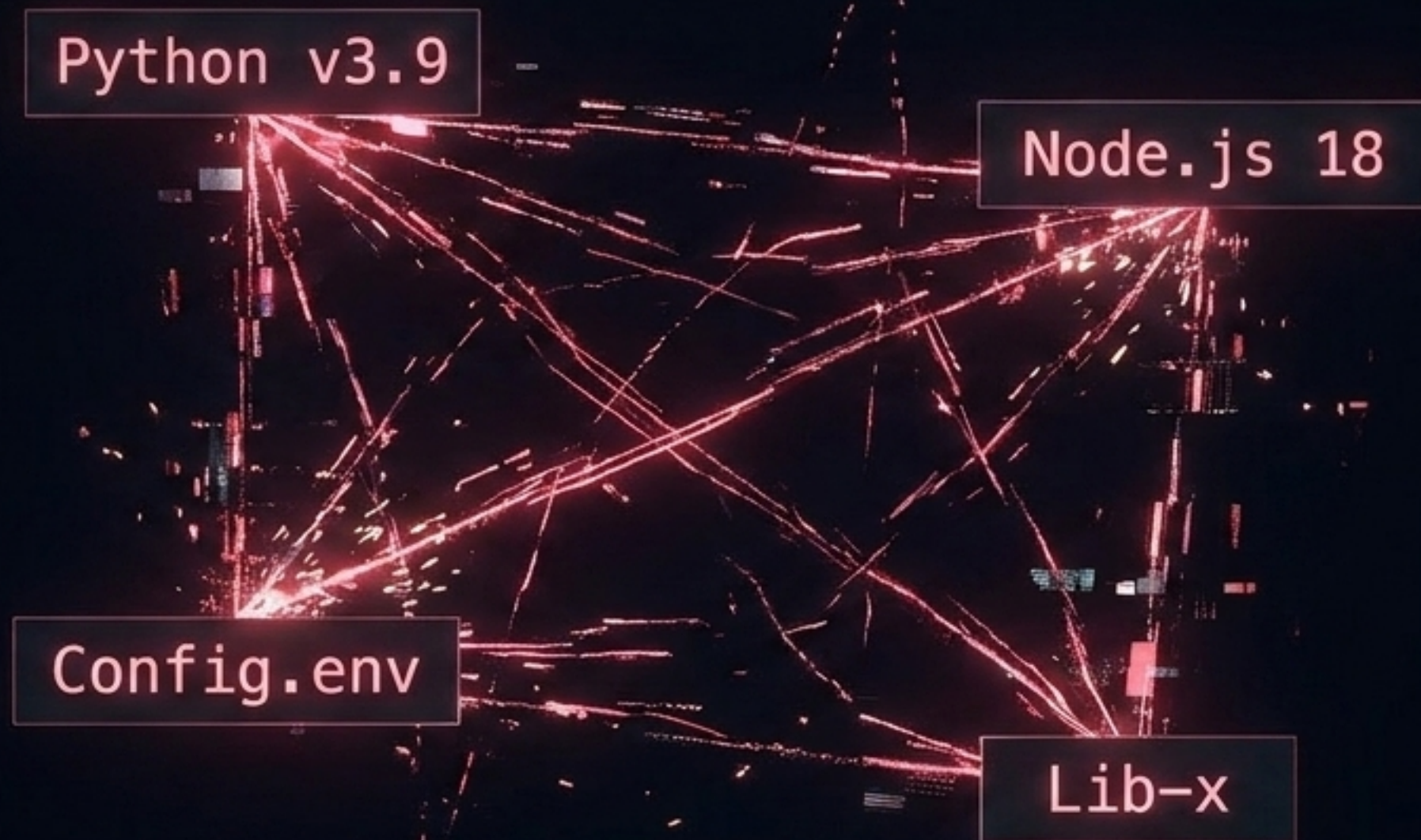
« روی سیستم من که کار می‌کنه! »

« وقتی برنامه‌ای روی لپ‌تاپ کاملاً درست کار می‌کند، اما روی سرور یا سیستم همکار شما با خطا مواجه می‌شود. »



«مشکل اصلی چه بود؟»

«هر برنامه به نسخه‌های خاصی
خاصی از زبان‌ها، کتابخانه‌ها و
تنظیمات نیاز دارد. تغییر فقط
یکی از این موارد، مساوی
است با اجرای ناموفق برنامه در
سیستم مقصد.»

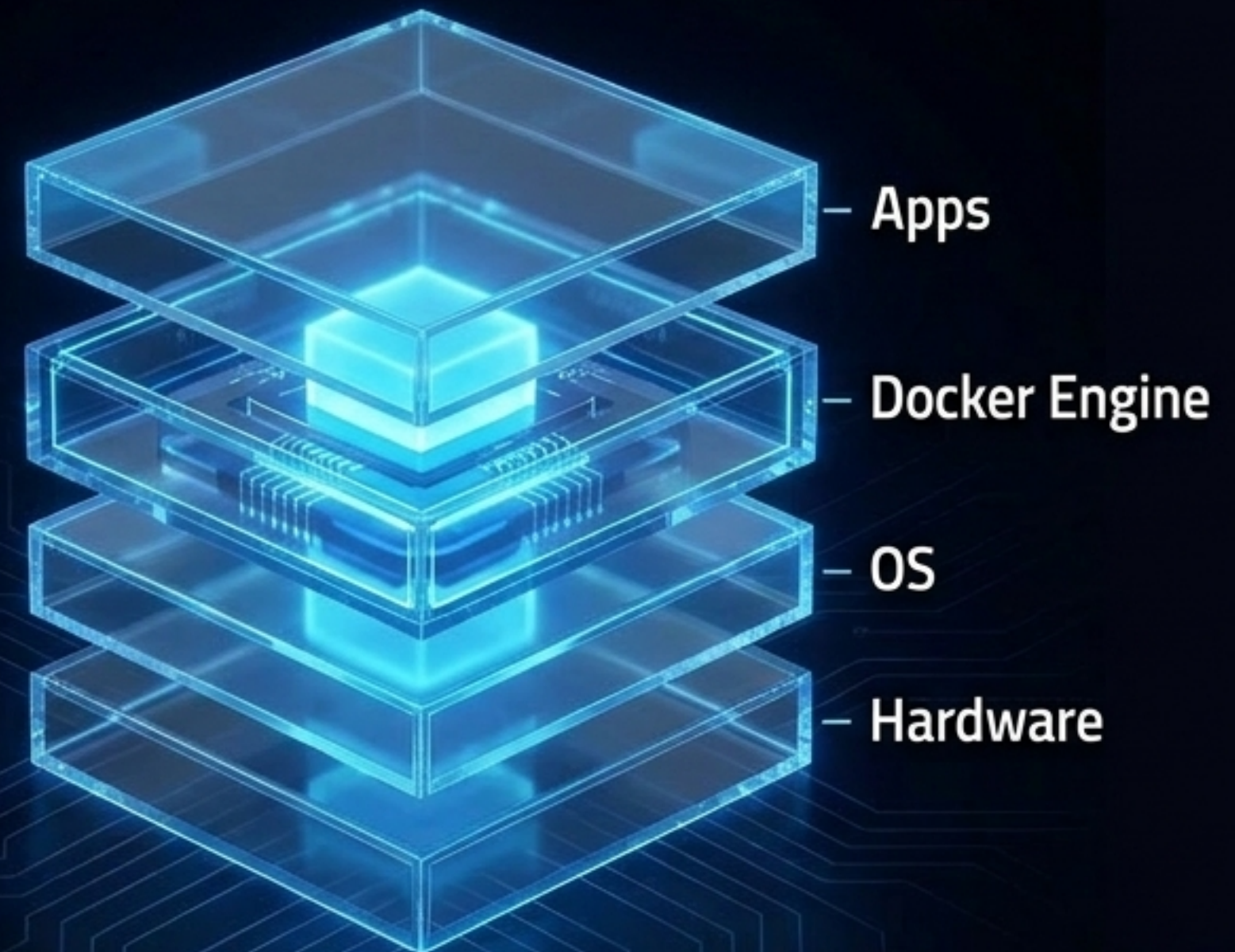


«کانتینر: یک جعبه هوشمند»

«همه چیز (برنامه، کتابخانه‌ها، تنظیمات) داخل این جعبه قرار می‌گیرد. این جعبه را به هر کامپیوتری ببرید، دقیقاً همان رفتار قبلی را خواهد داشت و دیگر به کامپیوتر مقصد وابسته نیست.»



«تفاوت ماشین مجازی و کانتینر»

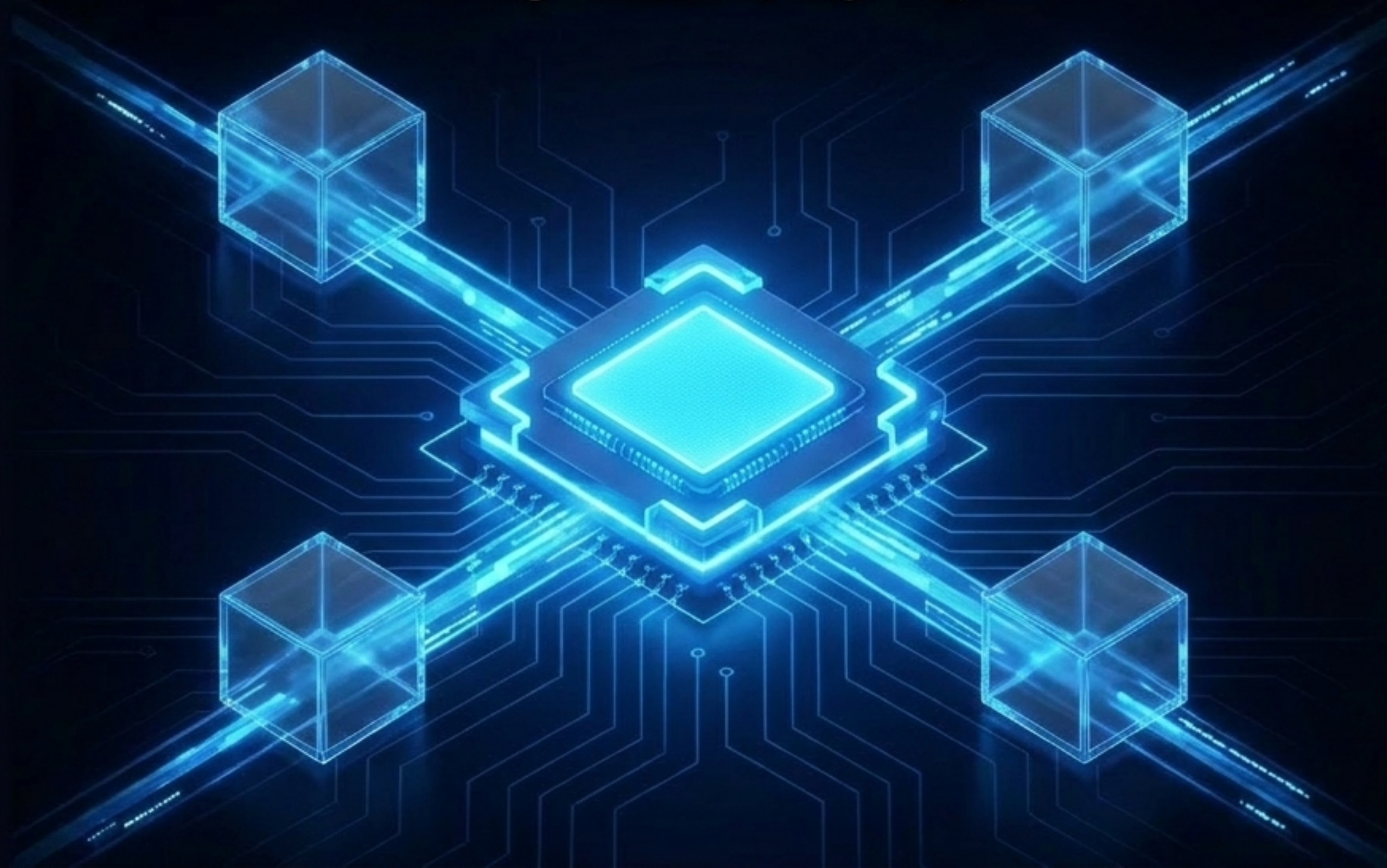


«ماشین مجازی نیازمند یک سیستم عامل کامل و مجزا است که حجم و زمان را هدر می دهد.
کانتینر از هسته میزبان استفاده می کند؛ بسیار سبک تر و سریع تر است.»



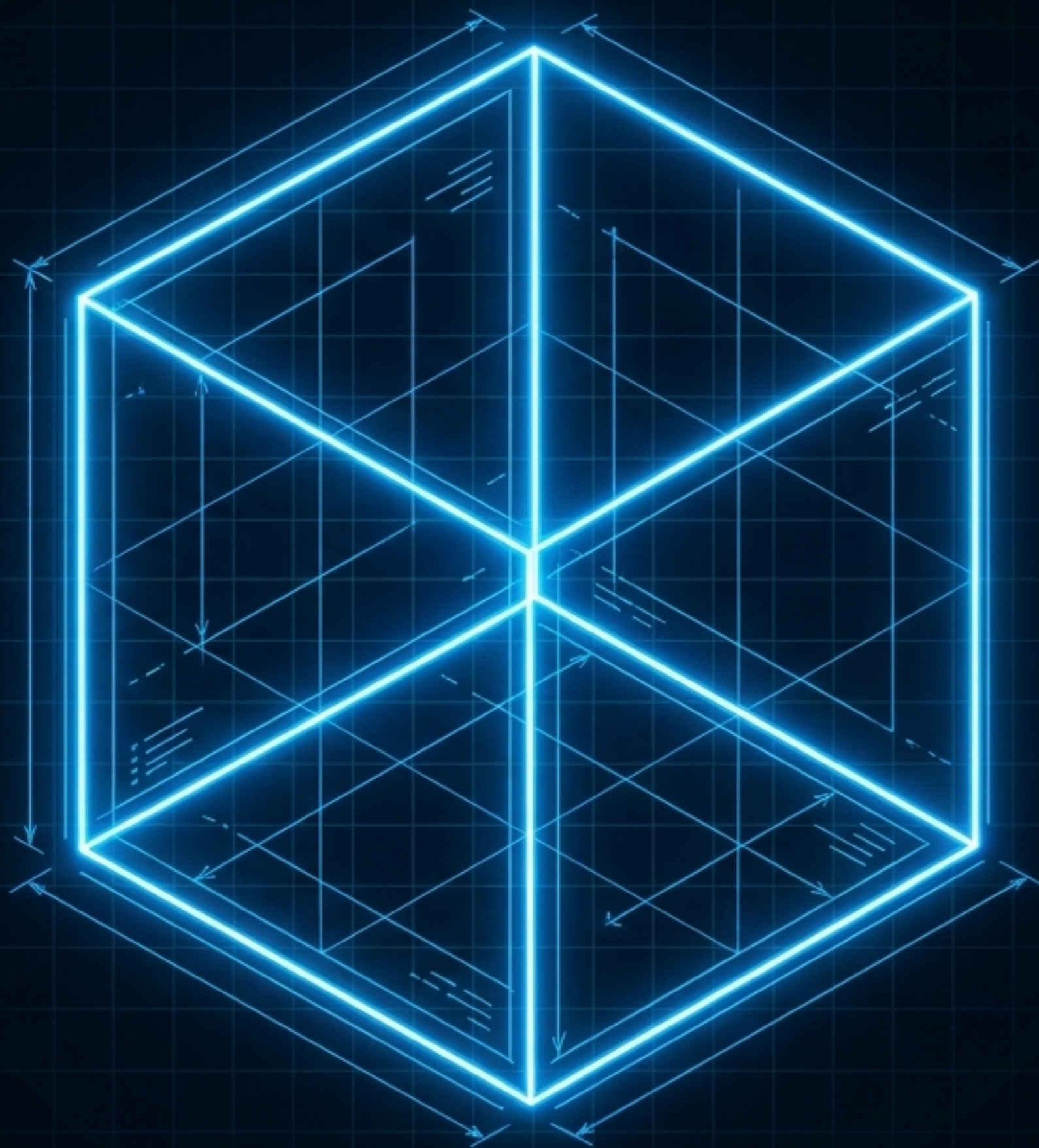
«پس Docker چیست؟»

«نرم‌افزاری که این کانتینرهای ایزوله را می‌سازد، اجرا می‌کند و به عنوان یک مدیر هوشمند، چرخه حیات آن‌ها را مدیریت می‌کند.»



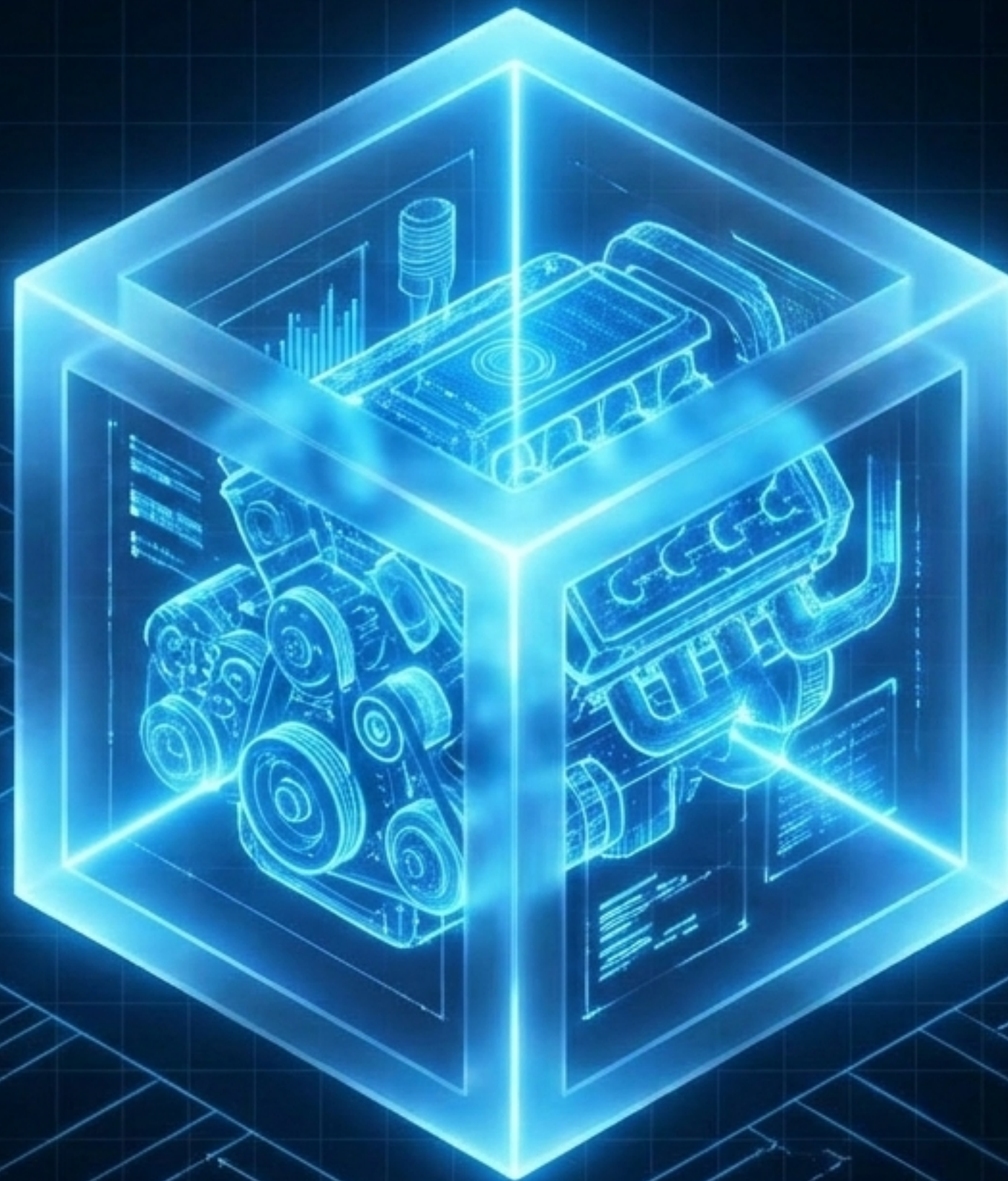
«مفهوم Image (نقشه ساخت)»

«قبل از ساختن یک خانه، نقشه آن وجود دارد. Image همان نقشه (فایل نصب) است؛ نصب) است؛ هنوز چیزی اجرا نشده، فقط مشخص شده که در صورت اجرا چه چیزی ساخته خواهد شد.»



«مفهوم Container (در حال اجرا)»

«کانتینر همان Image در حال اجراست.
درست مثل ویندوزی که نصب شده و کار
کر می‌کند. از یک Image واحد، می‌توان
صدها Container همزمان ساخت.»



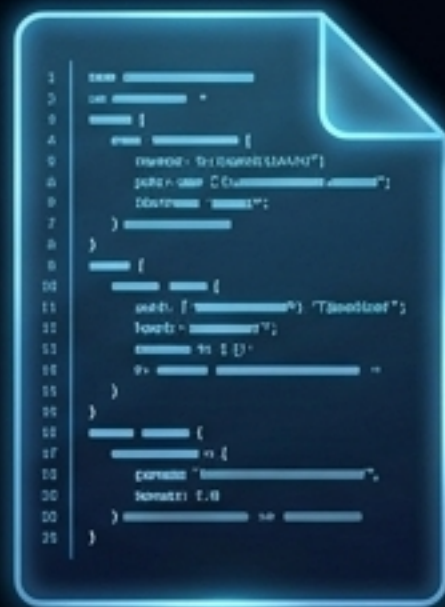
«دستور پخت: Dockerfile»

```
FROM  
COPY  
RUN  
CMD
```

«اگر Image یک کیک باشد، Dockerfile دستور پخت آن است. متنی که مرحله به مرحله مشخص می‌کند از چه سیستم‌عاملی شروع کنیم، چه فایل‌هایی را کپی کنیم و در نهایت چه دستوری اجرا شود.»



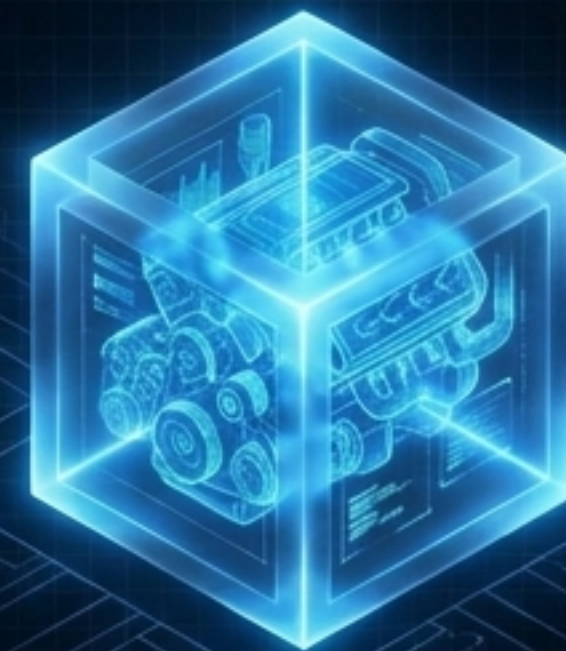
«فرآیند ساخت (Docker Build)»



«متن (Dockerfile)»



«موتور ساخت (Build Engine)»



«تصویر نهایی (Docker Image)»

فایل Dockerfile فقط یک متن است. با اجرای دستور Build، داکر این متن را خط به خط می‌خواند و در نهایت یک Image آماده‌ی استفاده تحویل می‌دهد.



«اکوسیستم Docker Hub»

«لازم نیست همه چیز را از صفر بسازید.
هزاران Image آماده و بهینه شده از
دیتابیس ها و سیستم عامل ها در این پایگاه
وجود دارد که فقط با یک دستور دانلود و
اجرا می شوند.»



«چالش پروژه‌های چندبخشی»

«پروژه‌های واقعی شامل چندین بخش مجزا هستند که هر کدام در یک کانتینر اجرا می‌شوند. اجرای تک تک این کانتینرها به صورت دستی، فرآیندی پیچیده، سخت و زمان‌بر است.»

Frontend



Backend



Database

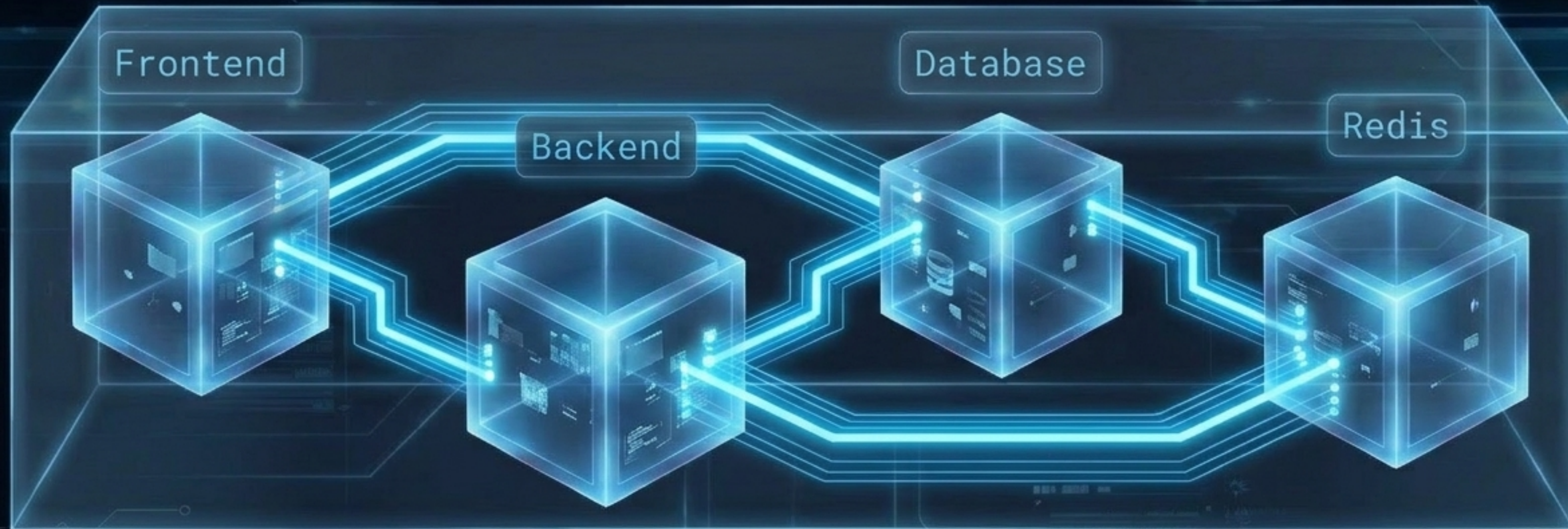


Redis



«مدیریت یکپارچه با Docker Compose»

«به جای اجرای تکی، تمام کانتینرهای پروژه را داخل یک فایل تعریف کنید تا کل سیستم تنها با یک دستور (docker compose up) به صورت یکپارچه و متصل راه اندازی شود.»



«شبکه‌سازی (Network)»

«کانتینرها در ابتدا مانند کامپیوترهای کاملاً ایزوله هستند. **داگر یک شبکه مجازی** خصوصی بین بین آنها می‌سازد تا بتوانند به صورت **امن و داخلی** با یکدیگر ارتباط برقرار کنند.»



«درگاه‌های ارتباطی (Ports)»

برنامه شما داخل کانتینر ممکن است روی پورت ۸۰ کار کند. داکر این پورت داخلی را به یک پورت در کامپیوتر میزبان (مثلاً ۸۰۸۰) متصل می‌کند تا کاربران خارجی بتوانند به آن دسترسی داشته باشند.

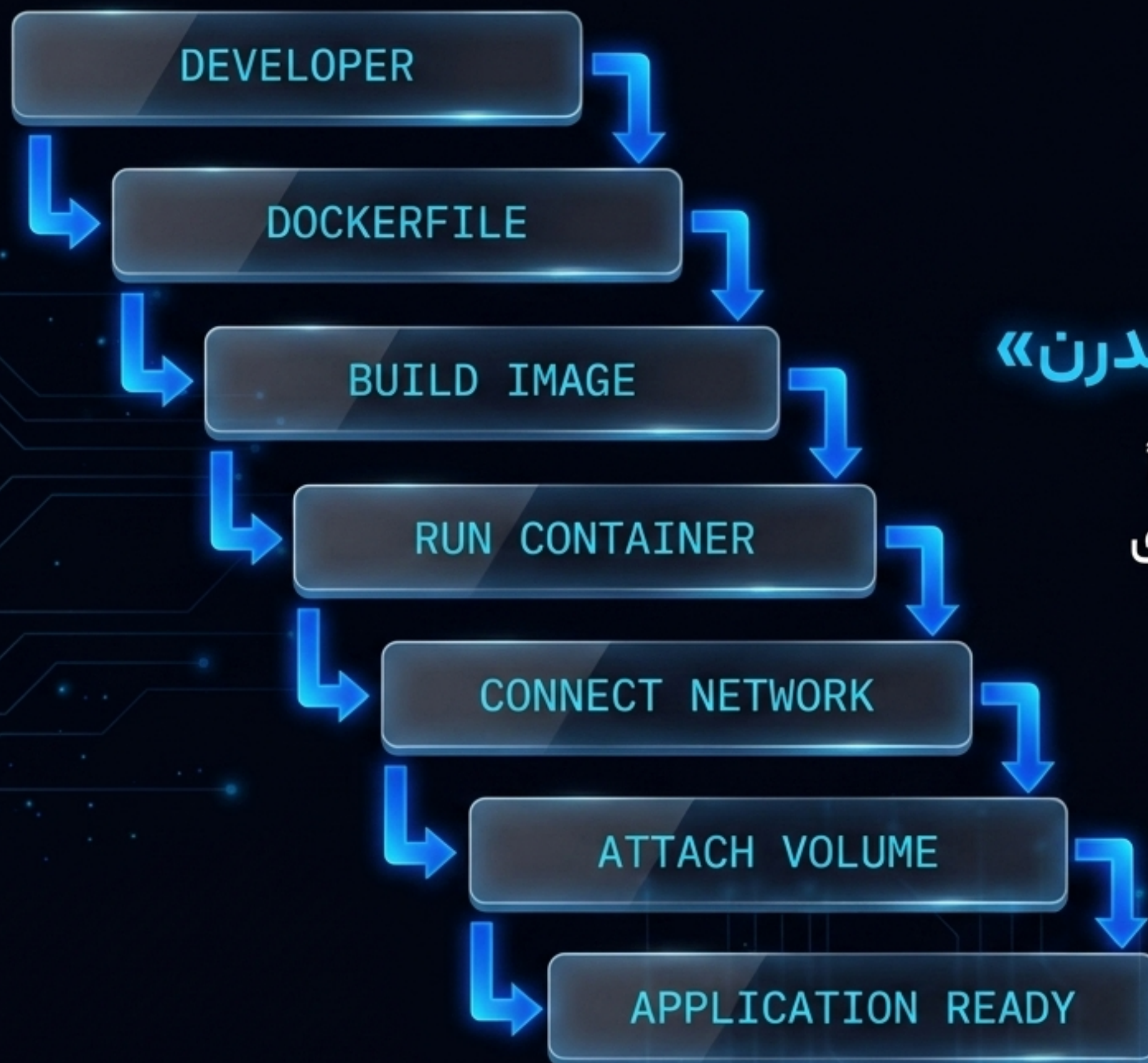
Port 8080

Port 80

«حفظ دائمی اطلاعات (Volumes)»

«ذات کانتینر موقتی است؛ با حذف آن، اطلاعات درونش نابود می‌شود. Volume یک هارددیسک مجازی و مجزاست که داده‌های حیاتی (مثل پایگاه داده) را حتی پس از حذف کانتینر برای همیشه حفظ می‌کند.»





«روند واقعی اجرای یک پروژه مدرن»

این مسیر استاندارد، چارچوبی است که تقریباً تمام پروژه‌های حرفه‌ای برای رسیدن از کدهای اولیه توسعه‌دهنده به مرحله اجرای نهایی و پایدار طی می‌کنند.



«اصول حرفه‌ای‌ها و اشتباهات تازه‌کارها»



CONTAINER

«اصل طلایی مهندسی: هر کانتینر باید فقط و فقط یک مسئولیت داشته باشد.»



«اشتباه مهلک: اجرای پایگاه داده بدون Volume و Volume و از دست دادن دائمی اطلاعات کاربران.»



«پایان دوران «روی سیستم من کار می‌کنه!»»

«داکر صرفاً یک ابزار توسعه نیست؛ پایه و اساس استقرار مدرن نرم‌افزار در شرکت‌های بزرگ است. قانون جدید این است: اگر داخل کانتینر کار می‌کند، در سرورهای غول‌پیکر هم بدون مشکل کار خواهد کرد.»





چاودان