## مجازی‌ساز چیست؟

توسط مجازی‌ساز می‌توان یک کامپیوتر فیزیکی با منابع مانند RAM و CPU را به چند قسمت بین کاربران تقسیم کرد که به آن‌ها به اصطلاح در کامپیوتر ماشین مجازی و در سرور، سرور مجازی می‌گویند، تمام این ماشین‌ها توسط مجازی‌ساز ایجاد می‌شوند به طوری که هر ماشین دسترسی به منابع مشخص خود را دارد و نمی‌تواند به ماشین دیگری دسترسی پیدا کند. هر ماشین می‌تواند سیستم عامل خود را داشته باشد و قسمت سخت‌افزار خود را کنترل کند و هر تغییر مورد نیازی بدون اینکه به سرور دیگر تاثیری ایجاد کند را اعمال کند.

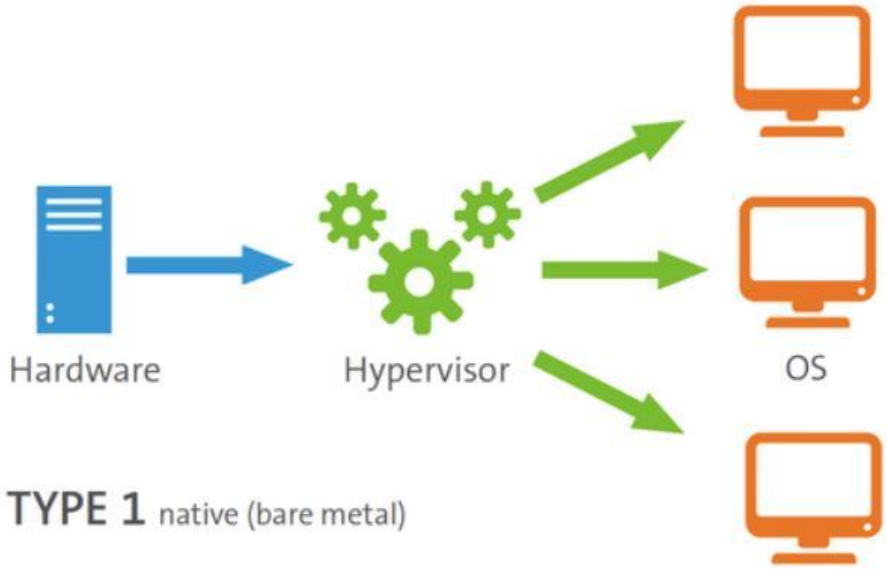
## Hypervisor چیست؟

یک Hypervisor یکی از دو روش برای مجازی‌سازی یک محیط محاسباتی است، منظور از virtualize، تقسیم منابع مانند RAM و CPU از یک محیط محاسباتی فیزیکی (به عنوان سرور اصلی میزبان) به چند ماشین مجازی کوچکتر(به عنوان مهمان) است. هر مهمان می‌تواند سیستم عامل مورد نیاز خود را نصب کند و هر ماشین مجازی منابع خود را دارد، در واقع سرور مجازی درست مانند یک سرور فیزیکی رفتار می‌کند، این امکان نیازمند پشتیبانی قابلیتی به نام VT-x در CPUهای intel و AMD-v در CPUهای AMD است.

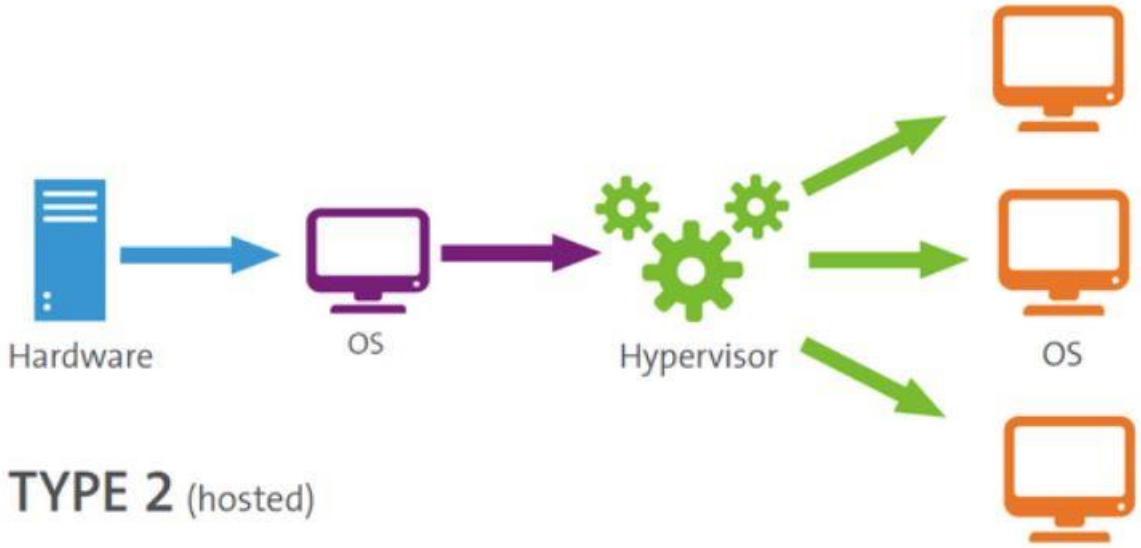
یکی از وظایف کلیدی که Hypervisor فراهم می‌کند جداسازی است، این به این معنی است که یک مهمان امکان دسترسی به سرور میزبان و همچنین دسترسی به سایر مهمان‌های ایجاد شده در سرور را ندارد و رفتارهای سرور مهمان روی آن‌ها تاثیری نداشته باشد، حتی اگر ماشین مهمان با مشکلاتی مانند کرش شدن مواجه شود. بنابراین Hypervisor باید به دقت مانند یک سخت‌افزار ماشین فیزیکی تقلید کند و از دسترسی مهمان به سخت‌افزار واقعی جلوگیری کند، از آن‌جایی که این عمل به شدت سرعت را کاهش می‌دهد از یک روش paravirtualized یا PV drivers استفاده می‌شود. این امکان تمام سخت‌افزارها را به صورت مجازی در اختیار ماشین مجازی قرار می‌دهد و درایورهای آن توسط Hypervisor دریافت می‌شود، با استفاده از این روش سرعت بالاتر می‌رود و همچنین امکان دسترسی مستقیم به سخت‌افزارهای اصلی سرور و کنترل آن‌ها توسط مهمان دیگر وجود ندارد.

Hypervisor دو نوع است:

- در این نوع از Hypervisor که به اصطلاح به آن "Bare Metal" گفته می‌شود، Hypervisor به طور مستقیم برای کنترل سخت‌افزار و سیستم عامل مهمان اجرا می‌شود.




- در این نوع از Hypervisor که به اصطلاح به آن "میزبانی شده" گفته می‌شود، Hypervisor در داخل یک سیستم عامل اجرا می‌شود و پس از آن سیستم‌عامل‌های مهمان ایجاد می‌شود. سیستم‌های مجازی دسکتاپ اغلب از این روش استفاده می‌کنند.



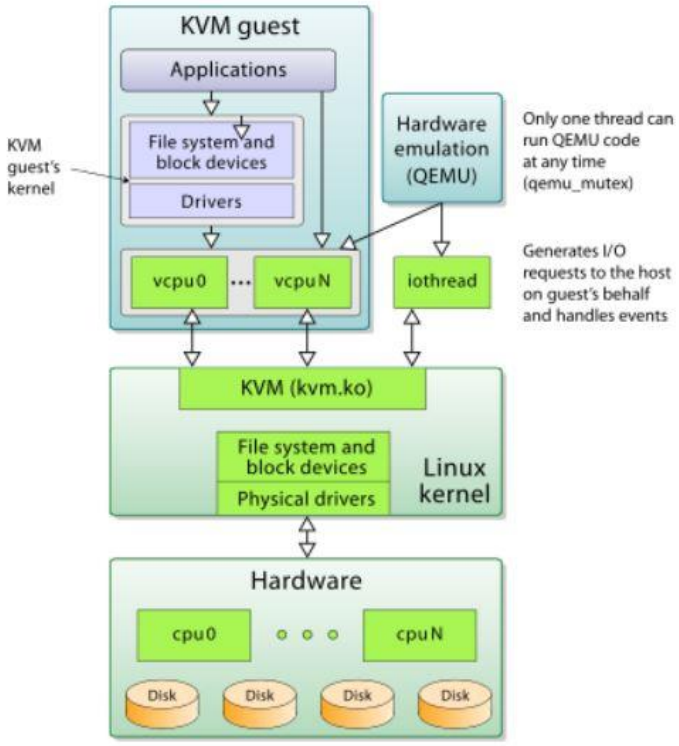
## تفاوت QEMU و KVM

- **مجازی‌سازی از نوع نمونه‌سازی:** نمونه‌سازی به معنای قابلیت تقلیدکردن یک نوع خاص از سخت‌افزار برای یک سیستم‌عامل صرف نظر از سیستم‌عامل زیرین است. برای مثال با استفاده از راه حل نمونه‌سازی می توان یک نسخه اسپارک از سیستم‌عامل سولاریس را روی یک کامپیوتر میزبان غیر اسپارک نصب کرد. نرم‌افزار نمونه‌سازی به عنوان یک کاربرد بر روی سیستم‌عامل میزبان اجرا می‌شود اما کل کامپیوتر بستر دیگری را نمونه‌سازی می‌کند. سیستم‌عامل مهمان هیچ گونه آگاهی از وضعیتش به عنوان یک سیستم‌عامل مهمان یا اینکه در یک محیط خارجی در حال اجرا است را ندارد. در بعضی از حالت ها، نمونه‌سازی سخت‌افزار می‌تواند کمی رنج‌آور باشد اما تکنولوژی‌های جدیدتر، نرم‌افزار و درایو های نمونه‌سازی را بروزرسانی کرده و پردازنده‌های میزبان ۶۴ بیتی سریع، نمونه‌سازی مجازی‌سازی را یک گزینه قابل دوام ساخته است. مخصوصا برای افرادی که نیاز به توسعه درایوها یا تکنولوژی‌ها برای بسترها دارند بدون اینکه بخواهند سرمایه‌گذاری زیادی در پشتیبانی پرسنل یا سخت‌افزار داشته باشند. بهترین مثال از نرم افزار نمونه سازی سخت افزار Bochs و QEMUمی‌باشد. QEMU یک نرم‌افزار نمونه‌سازی آزاد و متن باز است و بر روی تعداد محدودی از معماری‌های میزبان x86 و x64/x86 و power pc اجرا می‌شود اما نمونه‌سازی را برای معماری‌های دیگر از جمله ARM، MPIS و x86 و ... به عنوان سیستم‌عامل‌های مهمان انجام می‌دهد.
- **مجازی‌سازی در سطح کرنل:** مجازی‌سازی در سطح کرنل پدیده عجیبی در دنیای مجازی‌سازی است زیرا هر کرنل بدون توجه به کرنل در حال اجرا میزبان، از کرنل خاص خود برای بالا آوردن ماشین مجازی مهمان استفاده می‌کند. KVM لینوکس(ماشین مجازی کرنل) یک QEMU تغییر یافته است ولی برخلاف QEMU از گسترش‌های پردازنده مجازی‌سازی (AMD-v و INTEL-vt) استفاده می‌کند. KVM از سیستم‌عامل‌های مهمان زیادی با ساختارهای x86 و x86/x64 نظیر ویندوز و لینوکس پشتیبانی می‌کند. KVM از کرنل لینوکس به عنوان Hypervisor استفاده می‌کند و به عنوان یک ماژول قابل بارگیری کرنل اجرا می‌شود.

در واقع می‌توان گفت که QEMU از نوع دوم Hypervisor است که در فضای کاربر اجرا می‌شود. با استفاده از QEMU می‌توان معماری‌های مختلف و متفاوت از سرور را مجازی‌سازی کرد. اما KVM از نوع اول Hypervisor است که در فضای هسته اجرا می‌شود که به مراتب سریعتر عمل می‌کند اما بر روی همان معماری x64/x86 می‌توان مجازی‌سازی را انجام داد.

QEMU می‌تواند به طور مستقل اجرا شود، اما به دلیل شبیه‌سازی که به طور کامل در نرم‌افزار انجام می‌شود، بسیار کند است. در مورد خاصی که منبع و هدف هر دو معماری یکسان هستند(مانند مورد رایج x86 درx86)، همچنان باید کد را تجزیه کند تا هر «دستورالعمل ممتاز» را حذف کند و آنها را با سوئیچ‌های زمینه جایگزین کند. برای اینکه آن را تا حد ممکن در لینوکس x86 کارآمد کنید، یک ماژول هسته به نام KQemu وجود دارد که این کار را انجام می‌دهد. به عنوان یک ماژول هسته، KQemuقادر است اکثر کدها را بدون تغییر اجرا کند و تنها دستورالعمل‌های ring0 در پایین‌ترین سطح را جایگزین کند .در آن صورت، Userspace Qemuهمچنان تمام RAM را برای دستگاه شبیه‌سازی شده اختصاص می‌دهد و کد را بارگذاری می‌کند .تفاوت این است که به جای کامپایل مجدد کد، KQemuرا فراخوانی می کند تا آن را اسکن/پچ/اجرا کند .تمامی شبیه‌سازی سخت افزارهای جانبی در Qemu انجام می شود .این بسیار سریعتر از Qemu ساده است، زیرا اکثر کدها بدون تغییر هستند، اما همچنان باید کد ring0 را تبدیل کند (بیشتر کدهای موجود در هسته ماشین مجازی)، بنابراین عملکرد همچنان آسیب می بیند.

برای غلبه بر این مشکل، QEMU به شما این امکان را می‌دهد از KVM به عنوان شتاب‌دهنده استفاده کنید تا بتوان از پسوندهای مجازی‌سازی فیزیکی CPU استفاده کرد.



فایل اجرایی kvm-qemu مانند Qemu معمولی کار می‌کند RAM را اختصاص می‌دهد، کد را بارگذاری می‌کند و به جای کامپایل مجدد یا فراخوانیKQemu ، یک نخ ایجاد می‌کند، این رشته ماژول هسته KVM را فراخوانی می کند تا به حالت مهمان سوئیچ کند و به اجرای کد VM ادامه می‌دهد .در یک دستورالعمل ممتاز، به ماژول هسته KVM برمی‌گردد، که در صورت لزوم، به رشته Qemu سیگنال می‌دهد تا بیشتر شبیه‌سازی سخت‌افزاری را مدیریت کند.  یکی از چیزهای خوب این معماری این است که کد مهمان در یک رشته posix شبیه‌سازی شده است که می‌توانید با ابزارهای معمولی لینوکس آن را مدیریت کنید. اگر یک ماشین مجازی با ۲ یا ۴ هسته می خواهید، kvm-qemu ۲ یا ۴ رشته ایجاد می‌کند، هر یک از آنها ماژول هسته KVM را برای شروع اجرا فراخوانی می‌کند.  concurrency- اگر هسته های واقعی کافی دارید - یا scheduling - اگر نه - توسط زمانبند معمولی لینوکس مدیریت می‌شود، کد را کوچک نگه می‌دارد و شگفتی ها را محدود می‌کند.