archive.today
webpage capture

Saved from https://medium.com/@riteshgupta.ai/containers-vs-virtual-machines-a-develo search
no other snapshots from this url

All snapshots from host medium.com

29 Nov 2024 05:14:07 UTC

Webpage Screenshot share download .zip report bug or abuse

Member-only story

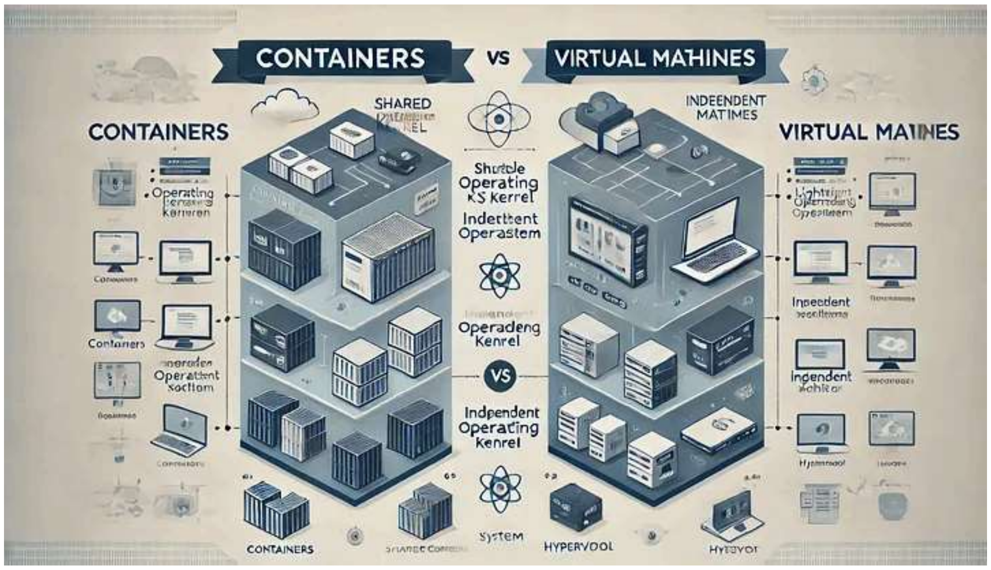
Containers vs. Virtual Machines: A Developer’s Guide

Ritesh Gupta · Follow
5 min read · Nov 11, 2024

10

Understanding When and How to Use Each

With the rise of cloud computing and the demand for scalable, efficient software deployment, developers often face the question: Should I use containers or virtual machines (VMs)? Both containers and VMs are used to package, deploy, and manage applications, but they achieve this in distinct ways, each with its own benefits, limitations, and use cases. In this guide, we’ll walk through the technical aspects of both technologies in a developer-friendly way, covering their architectures, security models, and resource management techniques.



1. The Basics: What Are Containers and Virtual Machines?

Containers and **Virtual Machines (VMs)** are both technologies used to virtualize and isolate applications, but they differ significantly in how they achieve this.

- **Containers:** Containers are lightweight, standalone units that package an application and its dependencies together. They run on a single host operating system and share its kernel, making them highly efficient and portable. The most popular container technology is Docker, though Kubernetes is also widely used for orchestrating containerized applications at scale.
- **Virtual Machines:** VMs are fully virtualized operating systems that include an application, its dependencies, and a complete OS kernel. VMs run on a hypervisor, which creates and manages these virtual environments on top of physical hardware. VMware, Microsoft Hyper-V, and KVM are some popular hypervisors.

2. Architecture Comparison

Containers Architecture

Containers use a shared operating system kernel and package only the application code and dependencies needed for the app to run. Here's a simplified view of how containers are structured:

- **Single OS Kernel**: Containers run on a single host OS kernel, sharing it across all containers. This is why they're lightweight — they don't need to replicate the entire OS environment.
- **Container Runtime**: Tools like Docker manage containers by providing a runtime environment. This runtime creates, manages, and executes containers on the host machine.
- **Images and Layers**: Containers use images (blueprints) and layered file systems, allowing images to be stored efficiently by reusing common layers across different containers.

Virtual Machines Architecture

VMs, on the other hand, are designed to provide complete isolation by emulating an entire operating system on top of a physical machine or a hypervisor. Here's what VM architecture looks like:

- **Hypervisor**: The hypervisor is the virtualization layer that sits between the physical hardware and the VM. It divides the hardware resources (CPU, memory, disk, etc.) and allocates them to each VM.
- **Guest OS**: Each VM runs its own guest OS, which can be different from the host OS. This full OS layer adds weight to VMs, making them larger in size and more resource-intensive.
- **Isolation and Independence**: VMs are isolated from each other and operate independently, which allows them to run completely different environments on the same physical hardware.

3. Resource Management and Efficiency

Containers: Efficient and Flexible

- **Lightweight**: Since containers share the host OS kernel, they're smaller in size and consume fewer resources compared to VMs. This efficiency enables rapid startup times, as there's no need to boot an OS each time.
- **Resource Sharing**: Containers can easily share resources, allowing more containers to run on the same hardware than VMs.
- **Scaling**: Containers are highly scalable; new instances can be created quickly, which makes them ideal for applications with varying loads.
- **Resource Limits**: Developers can set resource limits (CPU, memory) for containers to prevent a single container from hogging resources, although containers typically have looser isolation compared to VMs.

Virtual Machines: Robust but Heavy

- **Independent Resource Allocation**: VMs have their own OS, which allows them to manage resources independently of the host OS. This can make resource management straightforward, but it also means more overhead.
- **Dedicated Allocation**: Resources such as memory and CPU can be allocated specifically to each VM. Once allocated, these resources are reserved, which can limit flexibility.
- **Scalability**: VMs are slower to start up since they require OS boot time, but they are useful for long-running, stable applications where high isolation is crucial.

4. Security Considerations

Security is critical in both containers and VMs, but their different architectures lead to different security models.

Containers: Shared Kernel, Increased Risk

- **Kernel Vulnerabilities**: Since containers share the host OS kernel, a vulnerability in the kernel affects all containers. This shared kernel makes containers less isolated than VMs.
- **Isolation Techniques**: Container runtimes implement isolation through namespaces and control groups (cgroups), but this isolation isn't as robust as VMs.
- **Container Security Best Practices**:
- Run only trusted images and regularly update them to patch vulnerabilities.
- Use security scanning tools to detect vulnerabilities in container images.
- Implement runtime security policies to monitor and control container behavior.

Virtual Machines: Stronger Isolation

- **Complete Isolation**: VMs have their own OS, so they're more isolated from each other and the host OS, which limits the risk of cross-VM attacks.
- **Hypervisor Security**: Hypervisors add a layer of security, though they can be a potential point of attack. Modern hypervisors are generally secure, but they must be kept up-to-date.
- **VM Security Best Practices**:
- Regularly update both the host OS and guest OS with security patches.
- Use firewall and network segmentation to control access.
- Implement intrusion detection and monitoring systems for suspicious activities.

5. Use Cases: When to Use Containers vs. Virtual Machines

Understanding when to use containers and VMs can help developers design optimized, secure, and scalable applications.

Ideal Use Cases for Containers:

- **Microservices Architecture**: Containers are perfect for microservices since they allow developers to deploy, scale, and manage different parts of the application independently.
- **CI/CD Pipelines**: The fast startup times of containers make them ideal for continuous integration/continuous deployment (CI/CD), allowing teams

to test and deploy changes quickly.

- **Temporary and Lightweight Tasks:** Containers are well-suited for short-lived, lightweight tasks that don't require full OS-level isolation, such as batch processing or testing environments.

Ideal Use Cases for Virtual Machines:

- **Legacy Applications**: VMs are ideal for running legacy applications that require specific OS configurations or complete isolation from other applications.
- **Multi-tenant Environments**: VMs provide strong isolation, making them suitable for multi-tenant environments where different customers need complete security separation.
- **Hybrid Environments**: In cases where the host OS and guest OS need to be different (for example, running a Linux VM on a Windows server), VMs are the best solution.

6. Conclusion: Choosing the Right Technology

The choice between containers and virtual machines ultimately depends on the specific needs of the application. Containers are efficient, lightweight, and perfect for microservices and CI/CD pipelines, while VMs provide greater isolation and are ideal for legacy applications and multi-tenant environments.

In today's cloud-centric world, both technologies are commonly used together. Containers offer flexibility and speed, while VMs bring robust isolation. Developers can combine both by running containers inside VMs, gaining the advantages of both approaches in hybrid cloud or on-premises deployments.

By understanding the fundamental differences in architecture, security, and resource management, developers can make informed choices that optimize both performance and security for their applications.

Data Engineering Data Science Cloud Machine Learning Artificial Intelligence

10

Written by Ritesh Gupta

3.6K Followers · 28 Following

Follow

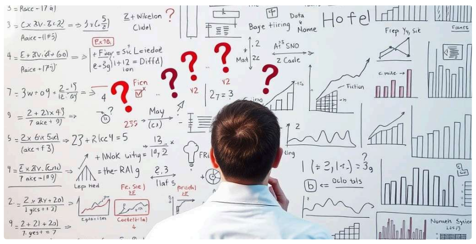
Data Scientist, I write Article on Machine Learning| Deep Learning| NLP | Open CV | AI Lover ❤️

No responses yet

What are your thoughts?

Respond

More from Ritesh Gupta



Ritesh Gupta

Can You Handle These 25 Toughest Data Science Interview Questions?

The role of a Data Scientist demands a unique blend of skills, including statistics, machine...

Sep 25 211 7

In Python in Plain English by Ritesh Gupta

7 GitHub Repos to Transform You into a Pro ML/AI Engineer

Hands-On Guides, Tools, and Frameworks to Fast-Track Your AI Journey

Nov 5 143 1

In Artificial Intelligence in Plain En... by Ritesh Gu...

Ritesh Gupta

From Jupyter to Production: Deploying Machine Learning...

Turn your Jupyter Notebook experiments into production-ready applications with this...

Oct 24 · 123

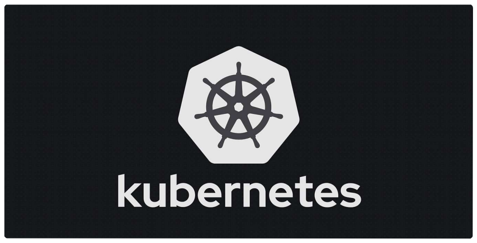
10 Must-Try LLM Projects to Boost Your Machine Learning Portfolio

🚀 10 Exciting LLM Projects to Elevate Your Machine Learning Skills! 🧠

Oct 6 · 124 · 1

See all from Ritesh Gupta

Recommended from Medium



In Stackademic by Crafting-Code

I Stopped Using Kubernetes. Our DevOps Team Is Happier Than Ever

Why Letting Go of Kubernetes Worked for Us

Nov 19 · 2.7K · 87

In ILLUMINATION by Sufyan Maan, M.Eng

A Simple System to Remember Everything You Read

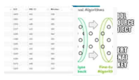
Simple yet effective

Nov 15 · 521 · 6

Lists

Predictive Modeling w/ Python

20 stories · 1691 saves



Natural Language Processing

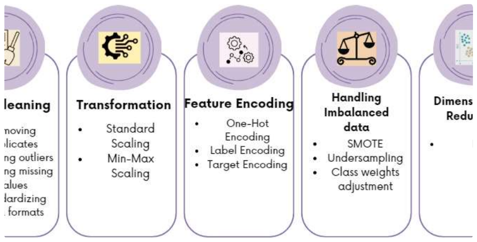
1839 stories · 1461 saves

Practical Guides to Machine Learning

10 stories · 2057 saves

ChatGPT prompts

50 stories · 2295 saves



In Code Like A Girl by Niveatha Manickavasagam

Top 5 Data Preprocessing Techniques: Beginner to...

A Comprehensive Guide to Clean, Transform, and Optimize Data Effectively

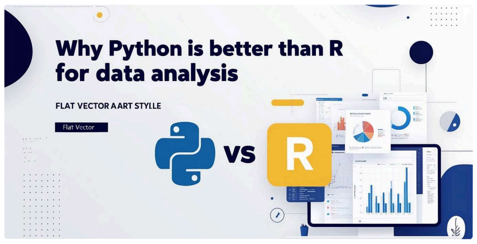
Nov 18 61 1

In Towards Data Science by Gustavo R Santos

Documenting Python Projects with MkDocs

Use Markdown to quickly create a beautiful documentation page for your projects

6d ago 455 4



In Python in Plain English by Mayur Koshti

Why Python is Better than R for Data Analysis

Python and R for data analysis would involve breaking down each relevant aspect, feature...

Nov 15 34 7

In DataDrivenInvestor by Bex T.

Firecrawl: How to Scrape Entire Websites With a Single Command...

And turn them into LLM-ready data

4d ago 294 3

See more recommendations