The Cloud

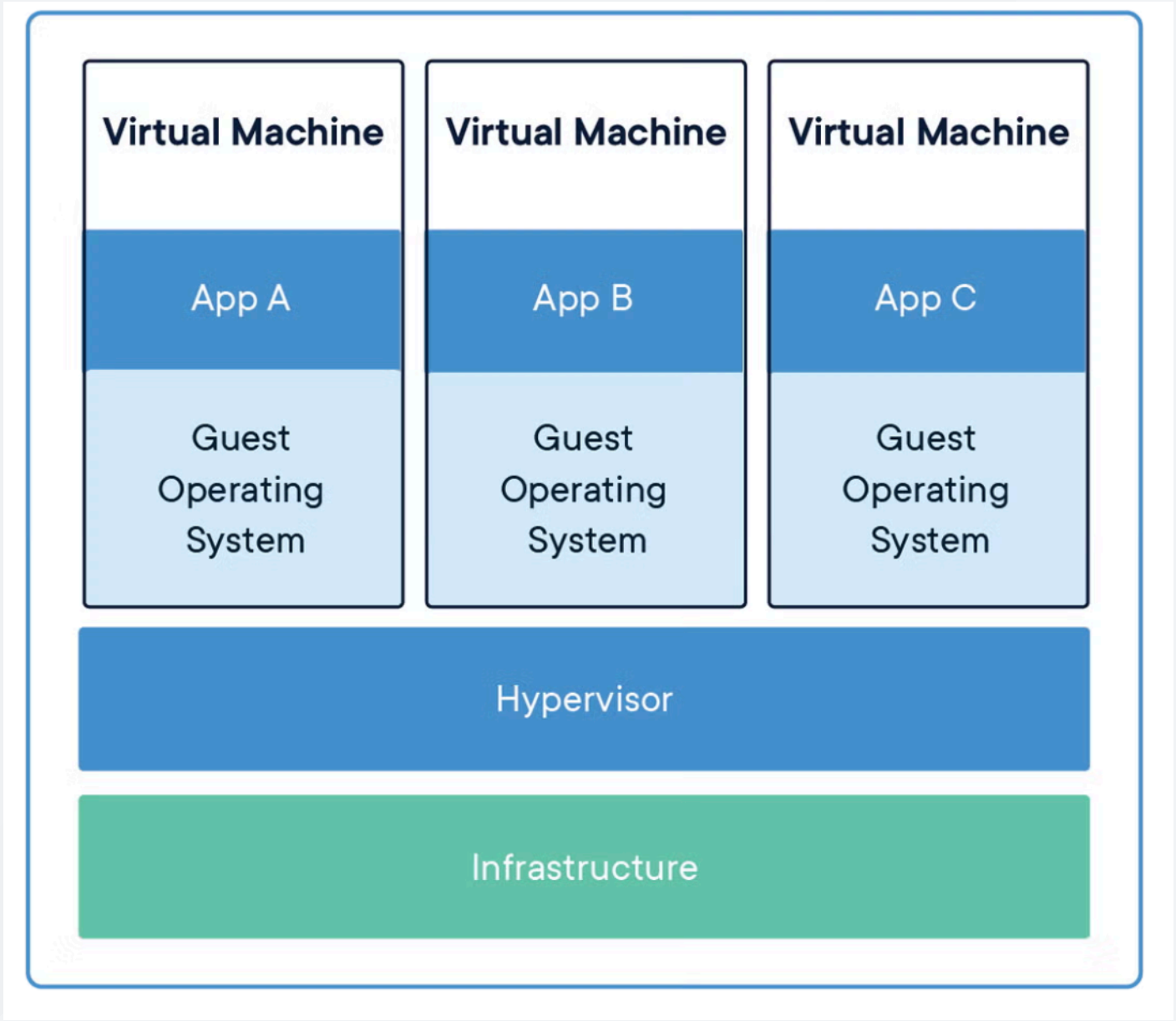
Agenda

1. Virtualization
2. Traditional Infrastructure
3. The Cloud
4. Managed Services
5. Cloud vs Local

Virtualization

Virtualization

- **Problem**: A single physical server running one application is not resource efficient
- **Solution**: Run multiple *virtual machines* (VMs) on one physical server
- A **hypervisor** (e.g. VMware, KVM) splits physical resources among VMs
- Each VM thinks it has its own dedicated hardware


Virtual Machine
App A
Guest Operating System
Virtual Machine
App B
Guest Operating System
Virtual Machine
App C
Guest Operating System
Hypervisor
Infrastructure

Benefits of Virtualization

- **Better hardware utilization**: Run multiple applications on one server
  - Can additionally run multiple applications on a single VM, this is what we use Docker/Kubernetes for!
- **Flexibility**: Easier to create/destroy VMs compared to physical servers
- **Snapshots**: Save and restore VM states
- **Isolation**: VMs are separated from each other

Virtualization

Virtualization does not solve everything. You still need to:

- Own and maintain the physical hardware
- Manage the hypervisor layer
- Handle capacity planning

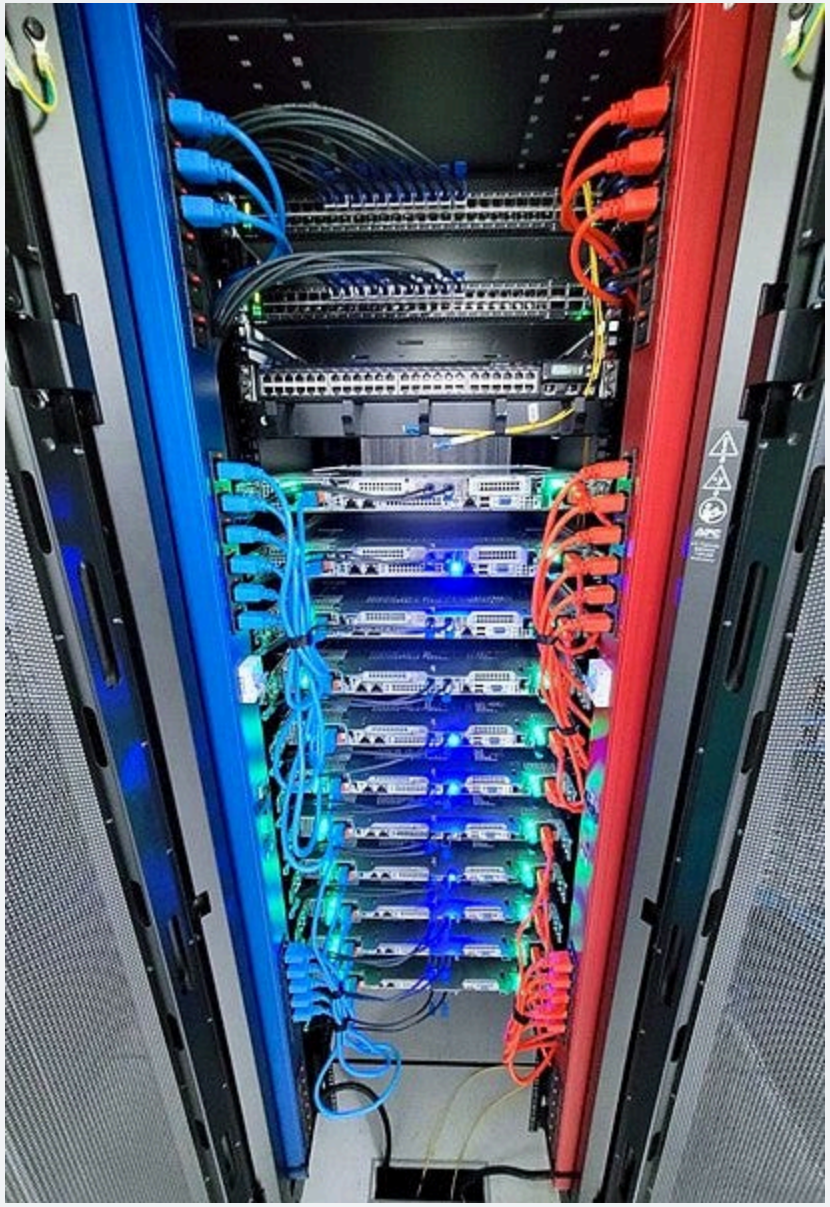
Traditional Infrastructure

Traditional Infrastructure

- Before the cloud, companies had to manage their own physical servers
- This meant purchasing, installing, and maintaining physical hardware in *racks*
- Required dedicated facilities (data centers) with:
  - Power and cooling systems
  - Physical security
  - Network infrastructure

What's in a Rack?

- **Servers**: Physical machines to run applications
- **Storage**: Hard drives/SSDs for data persistence
- **Networking**: Switches, routers, load balancers
- **Power**: UPS (Uninterruptible Power Supply) for redundancy
- **Cooling**: Prevent hardware from overheating



The Cost of Traditional Infrastructure

- **Upfront capital**: Expensive hardware purchases
- **Personnel**: System administrators to manage hardware
- **Space**: Physical data center space
- **Utilities**: Power and cooling costs
- **Maintenance**: Hardware failures, upgrades, replacements

Limitations of Traditional Infrastructure

- **Scaling is slow**: Ordering and installing new hardware takes weeks/months
- **Over-provisioning**: Must buy capacity for peak load, leading to waste during normal operation
- **Single points of failure**: Hardware failures can bring down entire services
- **Geographic constraints**: Limited to physical locations where you have data centers

The Cloud

What is "The Cloud"?

- Instead of owning hardware, rent *virtual machines* from a cloud provider (or bare metal)
- Major providers: AWS (Amazon), Google Cloud, Microsoft Azure, Alibaba
- They manage the racks and hypervisors, you manage your VMs and applications
- Pay only for what you use, can be short term or long term

The Cloud Trade-Off: Costs

Cloud Provider

- Pay per-use pricing
- No upfront capital
- Easy to scale
- Potentially expensive at scale

Your Own Sysadmin

- Fixed costs
- Large upfront hardware costs
- Hard to scale
- Predictable costs

The Cloud Trade-Off: Security

- **Advantage**: Professional security teams, compliance certifications, DDoS protection
- **Disadvantage**: No physical access to hardware
  - Must trust the provider, e.g. Amazon
  - Potential for data exposure
  - Shared infrastructure with other customers

The Cloud Trade-Off: Control

- **Advantage**: Don't need to worry about hardware failures, power outages, network issues
  - Still may want to have multi-cloud, i.e. AWS outage Oct 2025
- **Advantage**: Access to global infrastructure instantly
- **Disadvantage**: Limited control over underlying infrastructure
- **Disadvantage**: Vendor lock-in (hard to switch providers)

Managed Services

What are Managed Services?

- Cloud providers don't just rent bare servers
- They also offer *managed services*: pre-configured software that they operate for you
- Examples:
  - Managed databases (DynamoDB, RDS, Cloud SQL)
  - Managed Kubernetes (EKS, GKE)
  - Object storage (S3, Cloud Storage)

Two Types of Managed Services

Open Source

- Based on existing open source projects
- Examples: Managed PostgreSQL, Managed Kubernetes
- Can migrate off more easily

Proprietary

- Custom services built by provider
- Examples: AWS Lambda, DynamoDB, BigQuery
- Often more integrated

Open Source Managed Services

Pros:

- Familiar APIs and interfaces
- Can run locally for development
- Easier migration between providers
- Community support and documentation

Cons:

- May not integrate as deeply with provider's ecosystem
- Sometimes lag behind latest OSS versions

Proprietary Managed Services

Pros:

- Often better performance and integration
- Unique features not available elsewhere
- Deep integration with provider's other services

Cons:

- **Vendor lock-in**: Hard/impossible to migrate
- Hard to test locally
- Learning curve is provider-specific

Choosing Managed Services

- Consider your priorities:
  - **Portability** → favor open source services
  - **Performance/features** → proprietary may be worth it
  - **Team expertise** → stick with familiar technologies
- Hybrid approaches are common
  - Use proprietary services for non-critical components
  - Use open source for core business logic

Oct 20 AWS Outage

3:53 PM PDT Between 11:49 PM PDT on October 19 and 2:24 AM PDT on October 20, we experienced increased error rates and latencies for AWS Services in the US-EAST-1 Region. Additionally, services or features that rely on US-EAST-1 endpoints such as IAM and DynamoDB Global Tables also experienced issues during this time. At 12:26 AM on October 20, we identified the trigger of the event as DNS resolution issues for the regional DynamoDB service endpoints. After resolving the DynamoDB DNS issue at 2:24 AM, services began recovering but we had a subsequent impairment in the internal subsystem of EC2 that is responsible for launching EC2 instances due to its dependency on DynamoDB. As we continued to work through EC2 instance launch impairments, Network Load Balancer health checks also became impaired, resulting in network connectivity issues in multiple services such as Lambda, DynamoDB, and CloudWatch. We recovered the Network Load Balancer health checks at 9:38 AM. As part of the recovery effort, we temporarily throttled some operations such as EC2 instance launches, processing of SQS queues via Lambda Event Source Mappings, and asynchronous Lambda invocations. Over time we reduced throttling of operations and worked in parallel to resolve network connectivity issues until the services fully recovered. By 3:01 PM, all AWS services returned to normal operations. Some services such as AWS Config, Redshift, and Connect continue to have a backlog of messages that they will finish processing over the next few hours. We will share a detailed AWS post-event summary.

https://health.aws.amazon.com/health/status

Oct 20 AWS Outage: The DNS Bug

AWS uses automated "DNS Enactors" to update Route53 records for internal services. Two Enactors tried to update the DynamoDB endpoint at the same time (race-condition):

1. **Enactor A** starts an update but experiences unusual delays
2. **Enactor B** completes a newer update and cleans up old plans
3. **Enactor A** finally applies its now-outdated plan, overwriting Enactor B's
4. The cleanup process deletes this stale plan — removing **all IP addresses** from `dynamodb.us-east-1.amazonaws.com`

Result: an empty DNS record — every service in `us-east-1` that depended on DynamoDB lost connectivity

Oct 20 AWS Outage: Cascading Failures

1. DNS was restored ~3 hours later, but **cascading failures** followed:
    - EC2 couldn't establish VM leases without DynamoDB → "insufficient capacity" errors
    - Recovery attempts caused **congestive collapse** — too many retries at once
    - Network config propagation, load balancers, Lambda, ECS/EKS all fell over in sequence
2. Full recovery took **~14 hours** (11:48 PM → 2:20 PM next day)

Oct 20 AWS Outage: Takeaways

- A single DNS record deletion cascaded into a region-wide outage across dozens of services
- Even after the root cause was fixed, recovery was slow due to **thundering herd** effects
- A case for multi-cloud infrastructure, though distributing databases across providers is hard
  - Solutions exist (e.g. CockroachDB multi-cloud), but most companies don't do this
- Illustrates the **shared fate** of managed services — if your cloud provider's internal services depend on DynamoDB, so do you (even if you don't use it directly)

Lab: AWS