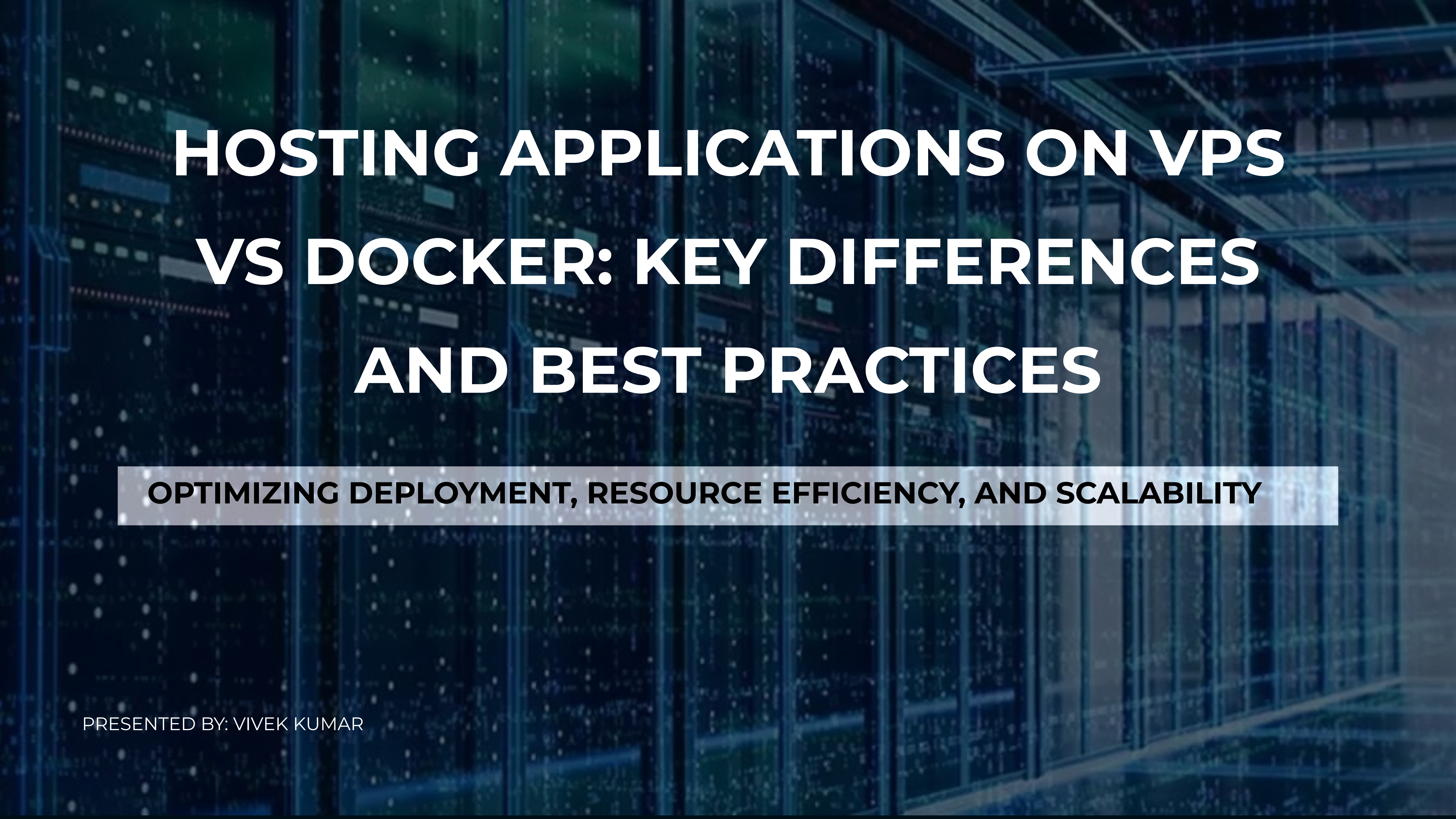


HOSTING APPLICATIONS ON VPS VS DOCKER: KEY DIFFERENCES AND BEST PRACTICES

OPTIMIZING DEPLOYMENT, RESOURCE EFFICIENCY, AND SCALABILITY

PRESENTED BY: VIVEK KUMAR

Abstract

Traditional VPS hosting vs. Docker's container-based approach

Case study:

Swoo App reduced AWS costs by 20% after migrating to Docker

Key focus:

Deployment, team productivity, cost efficiency, scalability

Kubernetes for managing containers efficiently

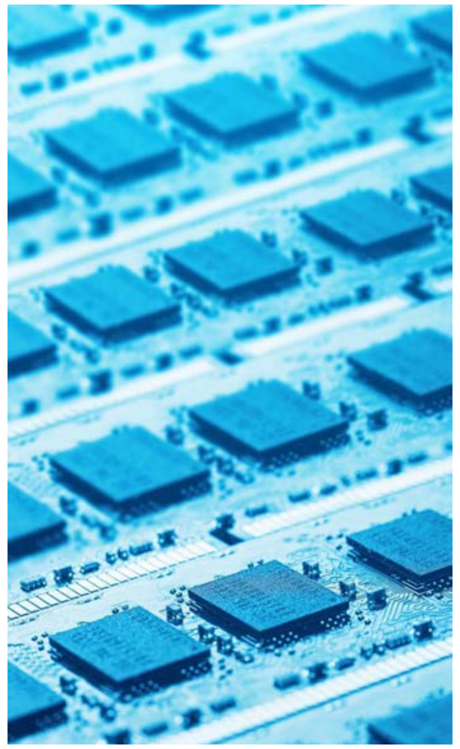
Agenda

1. Introduction to VPS and Docker
2. Deployment and Team Productivity
3. Application Management and Stability
4. Resource Efficiency and Cost Optimization
5. What is preferred: Docker or VPS?
6. Case Study: Swoo App Migration
7. Orchestration Tools: Kubernetes
8. Best Practices for Hosting Applications
9. Best Practices for Migrating to Docker

Introduction to VPS and Docker

VPS (Virtual Private Server)

- A virtualized environment on a shared physical server
- Provides dedicated resources (CPU, RAM, storage)



Docker (Containerization)

- Packages applications and dependencies into lightweight, portable containers
- Runs consistently across different environments

Key Differences: VPS vs Docker

Parameter	**VPS**	**Docker**
Isolation	Full OS per server	Process-level isolation
Resource Usage	High (full OS)	Lower (shared kernel)
Portability	Limited	High (runs anywhere)
Deployment Speed	Slower (manual)	Faster (automated)
Scalability	Manual scaling	Auto-scaling

Boot Time, Auto Scaling and Cost Optimization

Faster Boot Time

- Containers with micro or nano OS boot significantly faster than VPS or traditional VM
- Lightweight disk images allow containerized applications to start in **less than one second**, vs VPS partitions taking **3 to 10 seconds** to fully initialize

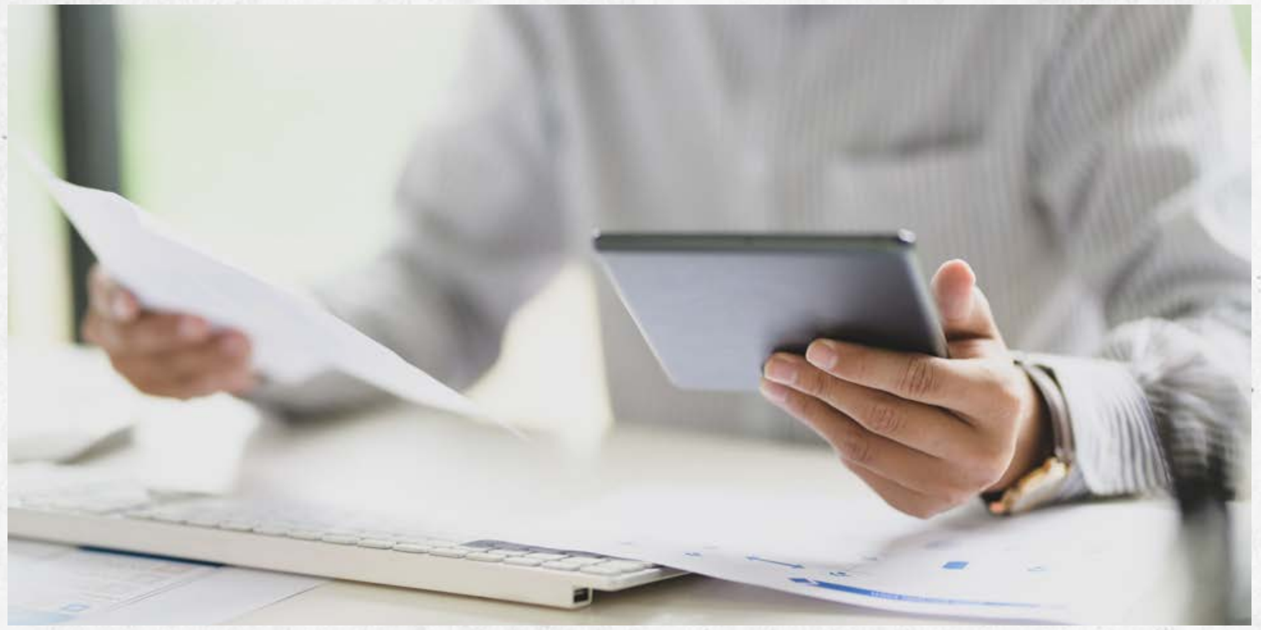
Optimized Resource Allocation

- Public cloud providers (AWS, Google Cloud, Azure) integrate Docker and Kubernetes tools to scale web/mobile apps dynamically without over-provisioning hardware
- System administrators can provision just the right amount of hardware for app support, reducing costs and improving efficiency

Elastic Scaling

- Disk images ensure fast container deployment to handle millions of users with personalized experiences

Deployment and Team Productivity



▶ VPS:

- Manual Environment Setup = **TIME-CONSUMING and ERROR-PRONE**
-

▶ Docker:

- Uses **Docker Images** for consistent environments
- Faster onboarding for teams
- Integrates with **CI/CD** pipelines for automation

Application Management and Scalability

VPS:

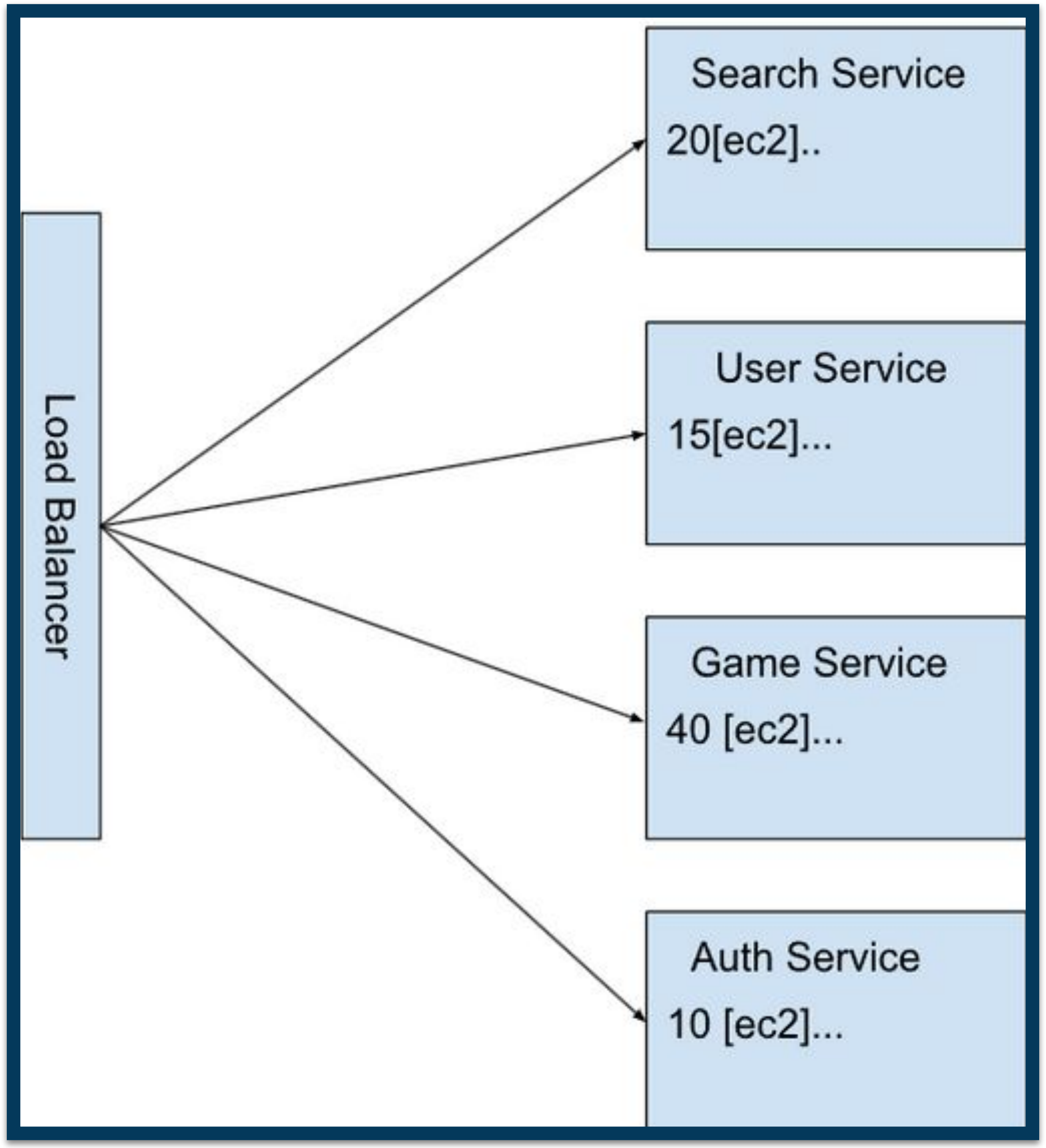
- Manual scaling and load balancing
- Complex management for multiple apps

Docker:

- **Easier scaling** with **Kubernetes and Docker Compose**
- Centralized app management

Case Study – Swoo App Migration

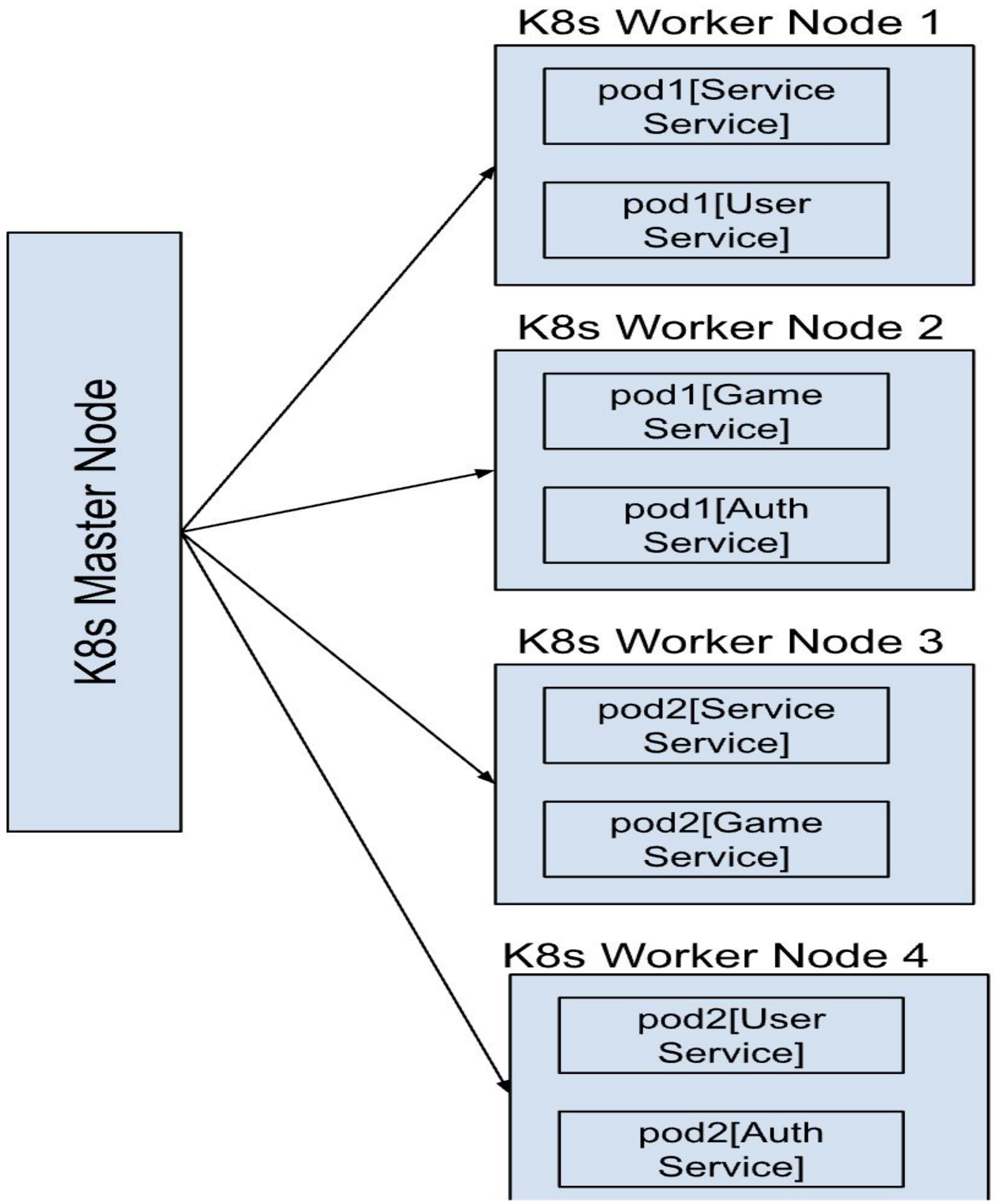
OLD ARCHITECTURE



- 85 EC2 instances were needed for Seach, User, Game and Auth Service
- It was served using API Gateway Kong via AWS autoscaling
- CPUs and Ram of EC2 was wasted due to underutilization

Case Study – Swoo App Migration

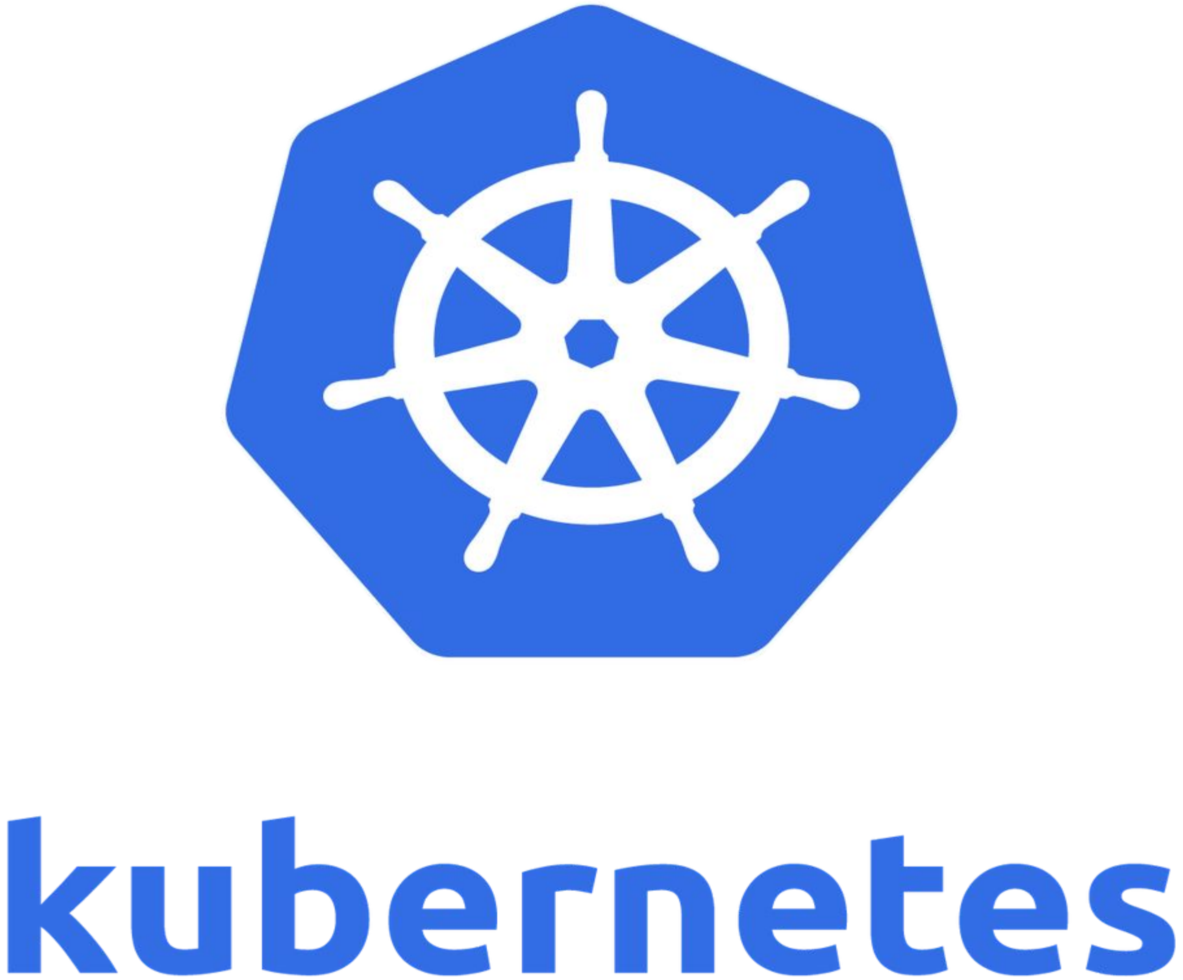
NEW ARCHITECTURE



- The migration of the Swoo app serves as a compelling case study showcasing the transformative impact of Docker and Kubernetes.
- Migration to Docker and Kubernetes, reduced the number of EC2 instances to 67 from 85,
- Led to significant reduction of 20% in AWS costs, demonstrating the efficiency and cost effectiveness of containerization and orchestration technologies.

Orchestration Tools: Kubernetes

Open-source platform for managing containerized applications



Benefits:

- Auto-scaling and self-healing
- Efficient resource allocation
- Simplifies container management

What is more preferred: VPS or Docker?

Key factors in decision making include

- **Project Management Approach** — The choice between VPS and container orchestration depends on team workflows, deployment strategies, and long-term maintainability.
- **Microservices Compatibility** — Both VPS and container platforms support web/mobile apps using microservices, but containers offer better service isolation and scalability
- **Web Traffic Scale** — The infrastructure choice is influenced by:
  - Page hits per day, determining server load
  - Simultaneous users, affecting real-time performance
  - Server and website caching configurations for speed optimization
  - Integrated CDN usage for faster content delivery
- **Budget and Development Capabilities** — Teams must consider infrastructure costs, scaling expenses, and the complexity of managing the chosen solution
- **Licensing Standards** — Open-source solutions like Docker and Kubernetes reduce vendor lock-in, while proprietary options may offer better enterprise support

VPS vs Docker Use Cases: Industry Adoption

Both VPS and container hosting can facilitate custom code requirements as well as distributed programming teams. Largely it depends on the expected or given user traffic base of a website, domain, or mobile app how much total hardware resources will be required to support operations in production.

Preferred by wide variety of web publishers, ecommerce websites, and multi-domain developers for their web hosting requirements

Preferred by corporate IT deployments in support of web/mobile applications like major media companies, finance/banking groups, industrial manufacturers, government organizations, etc. at scale in data center operations through elastic cluster web server networks

Best Practices for Hosting Applications

For VPS:

✔ Use configuration management tools (e.g., Ansible, Puppet).

✔ Monitor resource usage to avoid over-provisioning.

For Docker:

✔ Implement multi-stage builds to reduce Docker image size.

✔ Keep Docker images updated for security.

✔ Implement orchestration tools like Kubernetes

Best Practices for Migrating to Docker

🚀 **Assess and Refactor Microservices -** ensure all microservices are container-ready and can support Docker image builds

🔬 **Start Small and Iterate -** develop a small prototype before full-scale migration to identify potential issues early

🛠️ **Use Docker Compose for Testing -** for multi-container applications, leverage Docker Compose to streamline development and integration testing

🔄 **Integrate with CI/CD Pipelines -** automate deployments with Jenkins, GitLab CI/CD, or similar tools to ensure smooth rollouts

📦 **Orchestration with Kubernetes -** deploy and manage containers at scale using Kubernetes for better load balancing and service discovery

Best Practices for Migrating to Docker

🎓 **Train Your Team -** provide hands-on Docker and Kubernetes training to ensure smooth adoption

📊 **Implement Monitoring and Logging -** use tools like ELK Stack, Prometheus, and Grafana for real-time performance monitoring and debugging

⚡ **Thoroughly Test and Load Test -** conduct rigorous functional, integration, and load testing to ensure stability under high traffic

🌍 **Migrate Gradually with a Hybrid Approach -** use a phased migration strategy, running containers alongside existing infrastructure before full transition

Conclusion

Docker offers superior **efficiency**, **scalability**, and **automation** compared to traditional **VPS** solutions. When paired with **Kubernetes**, managing **large-scale containerized applications** becomes seamless and highly efficient. By migrating to Docker, organizations can achieve **cost reductions** while simultaneously enhancing **team productivity**, making it a powerful choice for modern application deployment and management.

THANK YOU.