COMPANY LIMITED
RACKSYNC
PERFORM

DEVOPS

INTERMEDIATE COURSE

Project Deploy & CI/CD

Optimizing Software Development Workflows

- Git & Source Control: Utilizing Git commands for version control and collaboration in Visual Studio Code.
- CI/CD Implementation: Understanding CI/CD concepts for scalable software delivery.
- GitHub Actions & GitLab CI: Automating workflows with GitHub Actions and GitLab for efficient development.
- Pipeline & Security: Configuring secure pipelines with CI/CD best practices.
- Project Automation: Enhancing efficiency with automation tools and deployment strategies.

Container Management

Optimizing Docker Deployments

- Docker Logging & Monitoring: Tracking container performance and troubleshooting issues.
- Resource Limitation: Managing CPU and memory constraints for optimized performance.
- Multi-Stage Builds: Reducing image size and improving security with multi-stage builds.
- Load Testing: Ensuring application resilience under heavy traffic.

Container Orchestration

Automation & Scalability

- Automation & Scalability: Automates deployment and scaling.
- Service Discovery & Load Balancing: Detects services and distributes traffic.
- Self-Healing: Automatically replaces failed containers.
- Declarative Configuration: Uses config files for a consistent state.
- Monitoring & Logging: Offers real-time insights for troubleshooting.

Reverse Proxy & CDN

Enhancing Web Application Performance & Security

- Nginx & Kong: Using Nginx and Kong as reverse proxies for load balancing and security.
- Cloudflare for DNS & CDN: Optimizing performance and security with Cloudflare’s DNS and CDN services.
- Service Routing: Efficiently directing traffic to backend services in web applications.
- Multi-Routing: Implementing advanced routing strategies for redundancy and failover.

Monitoring

Ensuring System Health & Performance

- Prometheus Metrics: Collecting and analyzing server data for health monitoring.
- Uptime Monitoring: Detecting service failures and sending alerts.
- Grafana Dashboards: Visualizing metrics for better decision-making.

Key Takeway

Review of DevOps Best Practices

CI/CD Optimization

CI/CD Optimization: Automating workflows with GitHub Actions and GitLab CI.

Proxy & CDN

Reverse Proxy & CDN Benefits: Enhancing performance and security with Nginx and Cloudflare.

Container Management

Container Management: Utilizing Docker and Kubernetes for scalable deployments.

Monitoring & Observability

Leveraging Prometheus and Grafana for real-time insights.

Scalability & Security

Ensuring applications are scalable and secure with modern tools.

Continuous Improvement

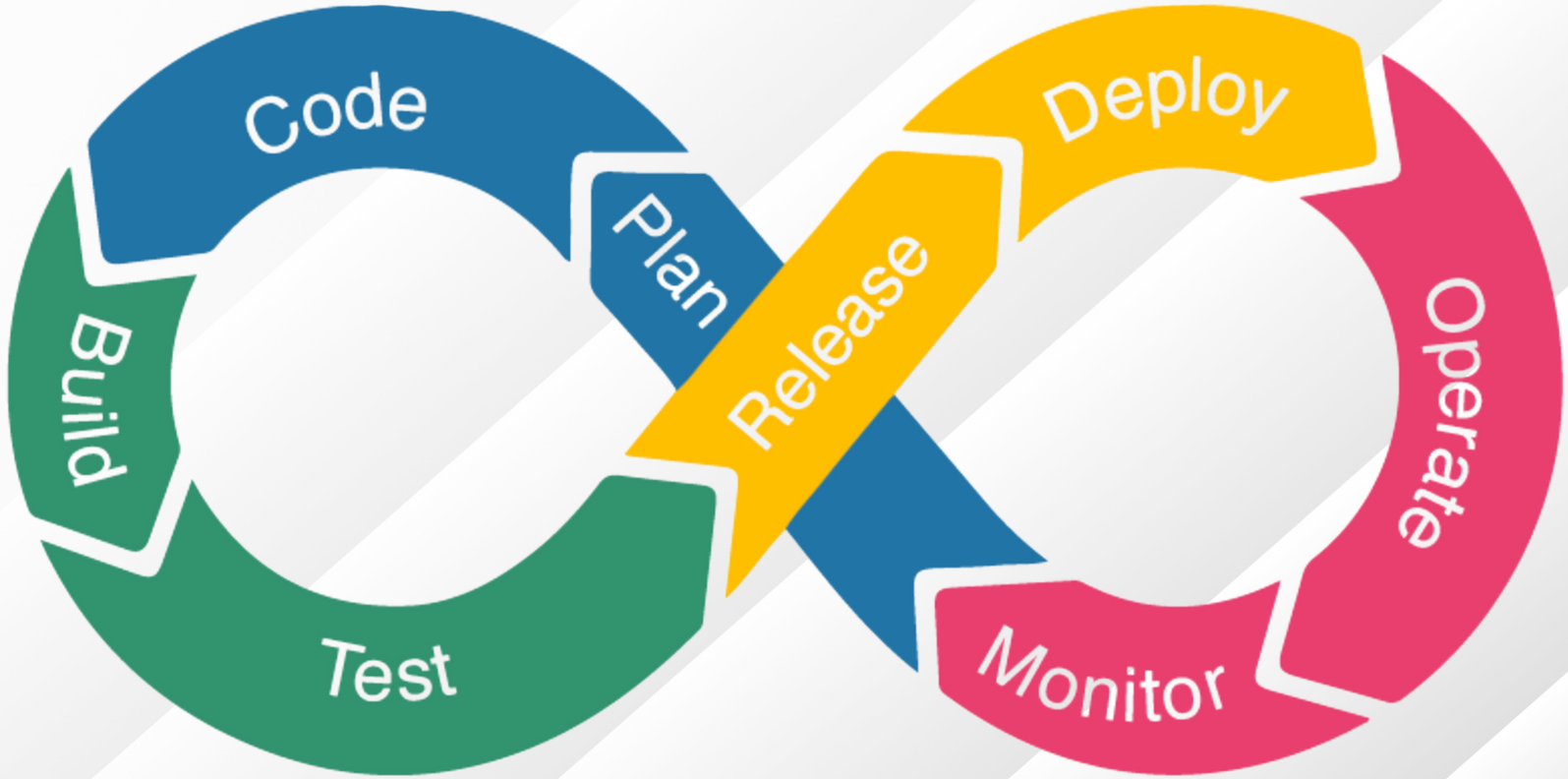
Using monitoring insights to refine and optimize systems.

What is DevOps?

- **DevOps** is a culture and practice that bridges the gap between development (Dev) and operations (Ops) teams
- **Goal:** Increase software delivery speed, improve quality, and reduce operational failures
- **Evolution:** From Waterfall → Agile → DevOps

Not just tools but a cultural and process transformation

DevOps Lifecycle



Why is DevOps Important?

Traditional Problems:

- Ineffective communication between development and operations teams
- Long delivery cycles (months/years)
- Risky and error-prone deployments
- "Works on my machine, but not on yours"

Benefits of DevOps:

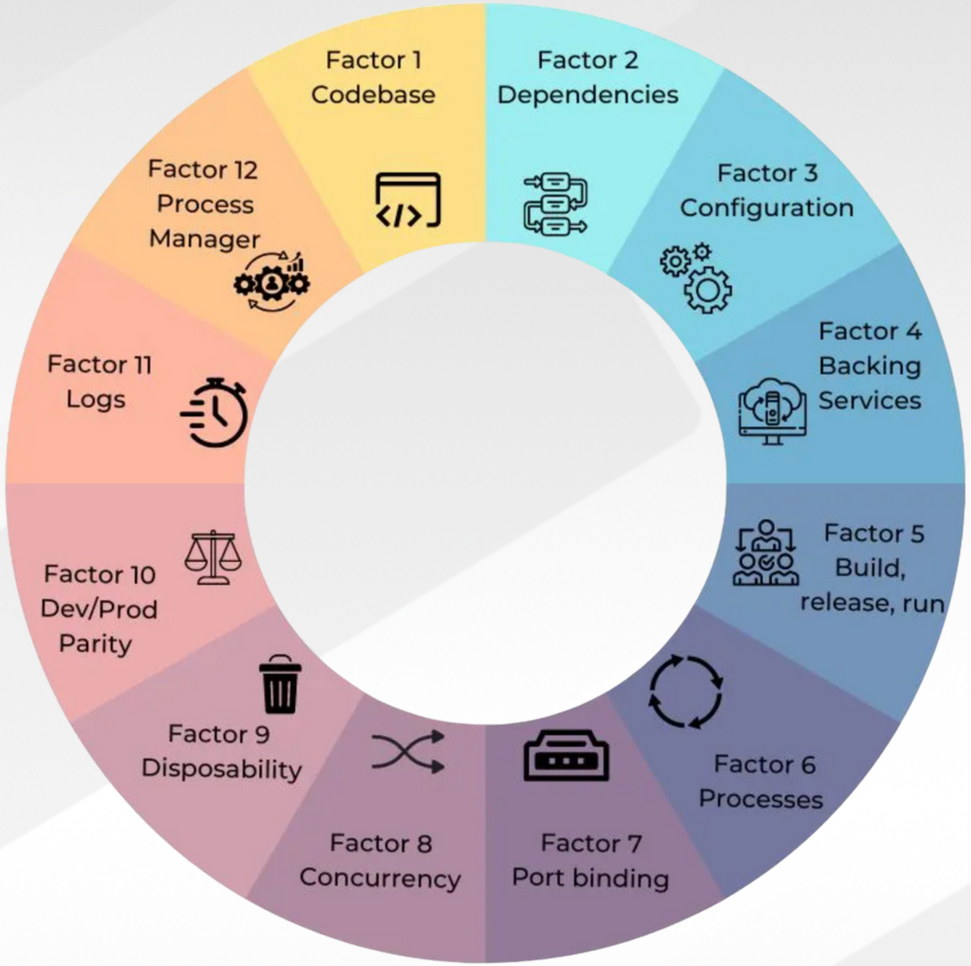
- Reduced lead time (from months to days or hours)
- Increased deployment frequency
- Lower change failure rate
- Reduced Mean Time To Recovery (MTTR)

DevOps Principles

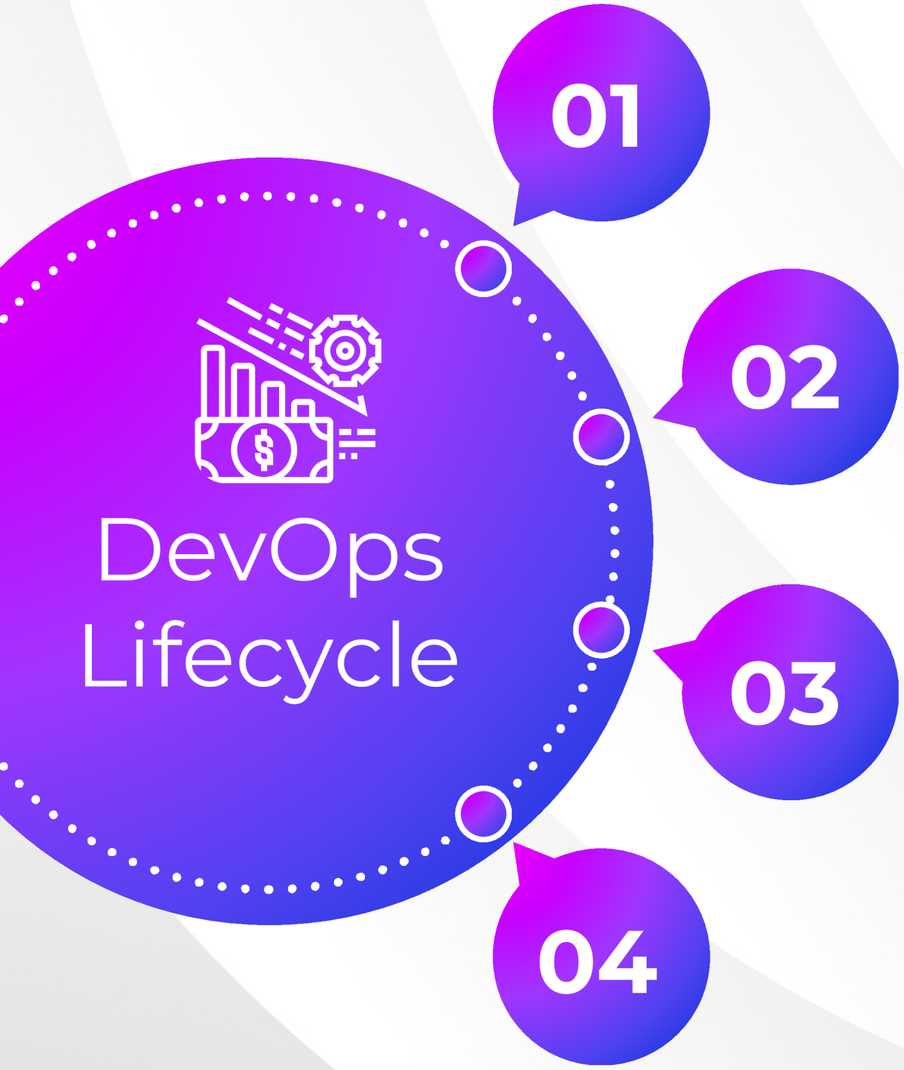
CAMS Model:

- **Culture** - Collaborative working culture
- **Automation** - Process automation
- **Measurement** - Measurement for improvement
- **Sharing** - Knowledge and responsibility sharing

The Twelve-Factor App



- **Codebase** - One codebase tracked in version control, many deployments
- **Dependencies** - Explicitly declare and isolate dependencies
- **Config** - Store config in the environment
- **Backing Services** - Treat backing services as attached resources
- **Build, Release, Run** - Strictly separate build and run stages
- **Processes** - Execute the app as one or more stateless processes
- **Port Binding** - Export services via port binding
- **Concurrency** - Scale out via the process model
- **Disposability** - Maximize robustness with fast startup and graceful shutdown
- **Dev/Prod Parity** - Keep development, staging, and production as similar as possible
- **Logs** - Treat logs as event streams
- **Admin Processes** - Run admin/management tasks as one-off processes



DevOps Infinity Loop:

- **Plan** - Planning and prioritizing
- **Code** - Code development
- **Build** - Building and testing code
- **Test** - Automated testing
- **Release** - Preparing for deployment
- **Deploy** - Deploying to production
- **Operate** - Operations and monitoring
- **Monitor** - Monitoring and data collection

CI/CD Basics

Continuous Integration (CI):

- Integrate code frequently (at least daily)
- Automated testing on every commit
- Detect issues earlier

Continuous Delivery/Deployment (CD):

- Delivery: Always ready to deploy (human pushes button)
- Deployment: Every validated change automatically deployed

Pipeline is the automated sequence from code to production

- We will learn how to build and customize pipelines in the next lesson

Thank you

RACKSYNC COMPANY LIMITED

Address : 135/5 Moo.1, Khuntalae Sub-District, Muang District, Surat Thani, 84100

info@racksync.com

https://racksync.com

FB : https://www.fb.com/racksync

Tel : 08 5880 8885