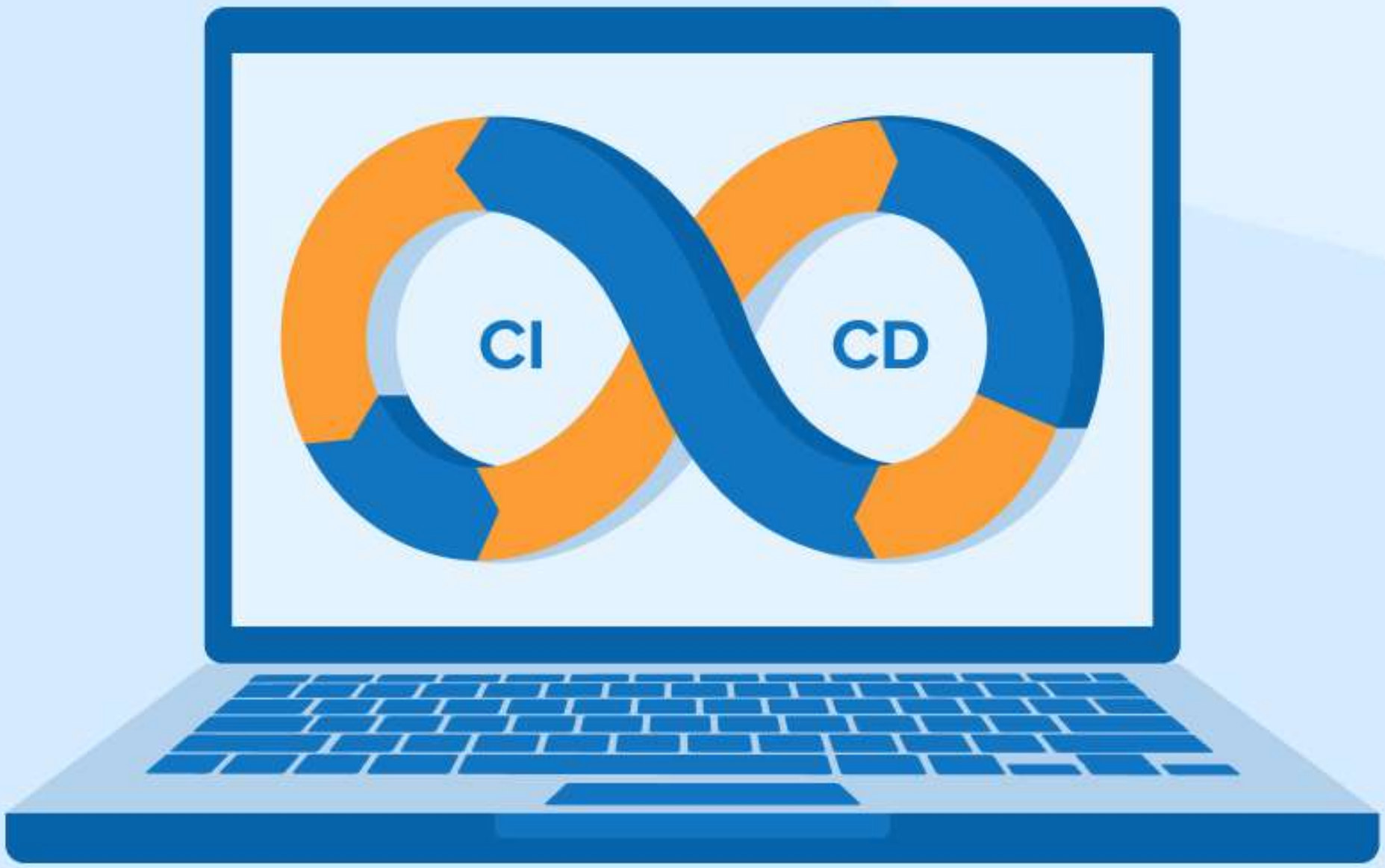
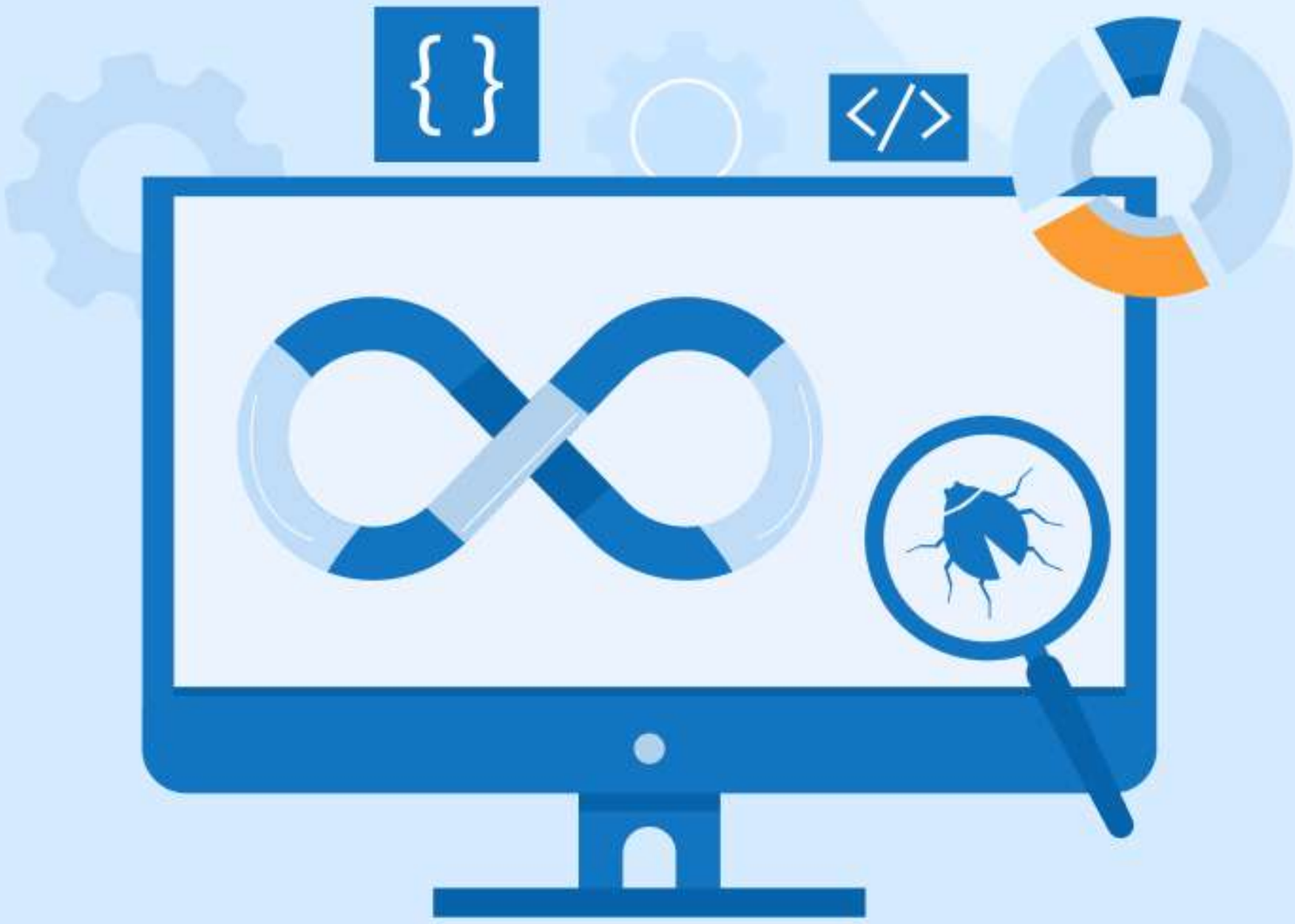


Key CI/CD Best Practices for Smooth Automation Process

TatvaSoft
sculpting thoughts...

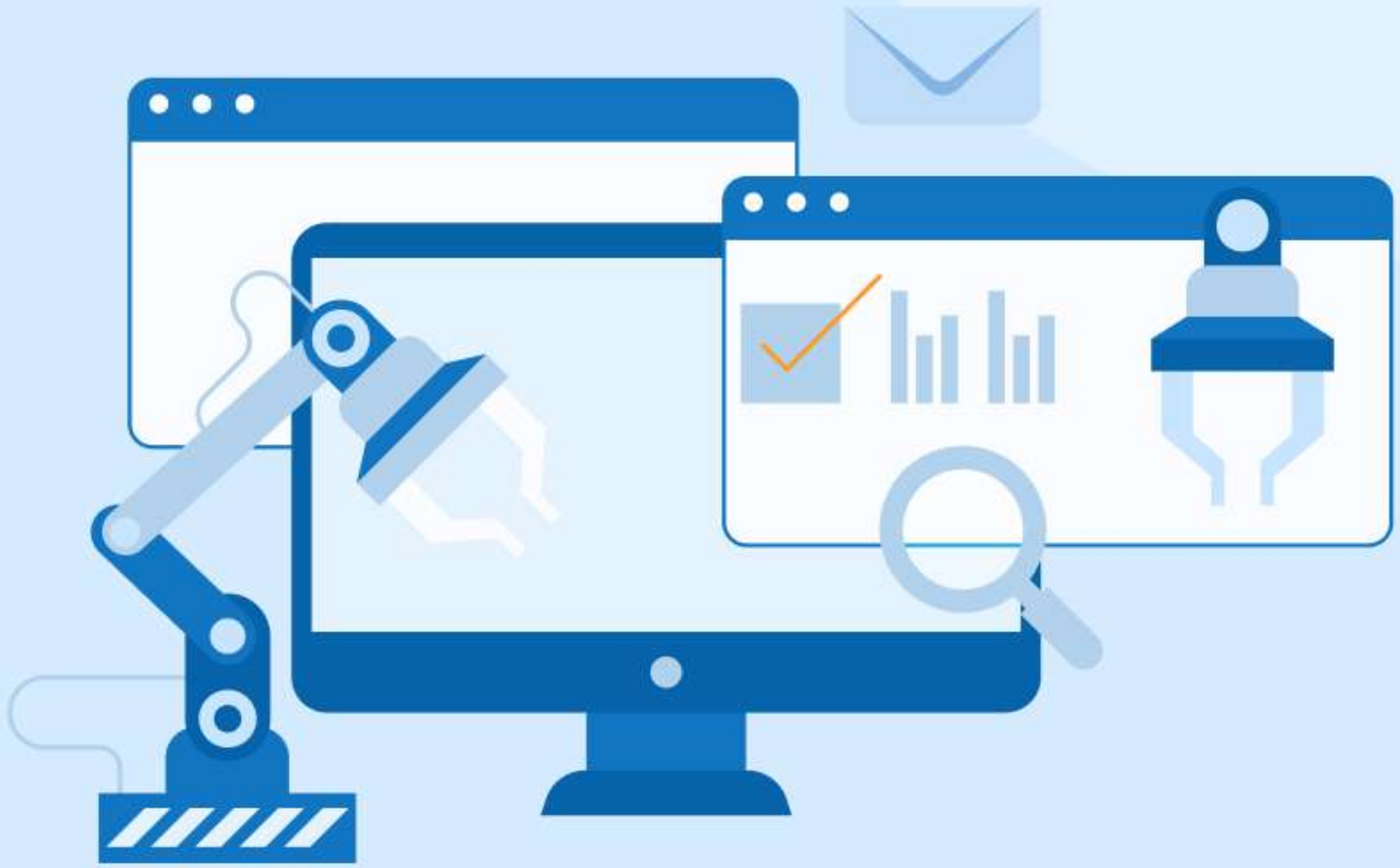


Commit Early, Commit Often

- Frequent updates enable automated testing and quick feedback for developers.
- Early and frequent commits help identify bugs faster and improve code quality.
- Regular commits prevent code loss and clarify deployment versions.

Prioritize Security

- Security is vital to prevent breaches in CI/CD pipelines.
- Automated security checks help quickly identify and fix vulnerabilities.
- Key measures include automated testing, access management, and network isolation.



Build only Once

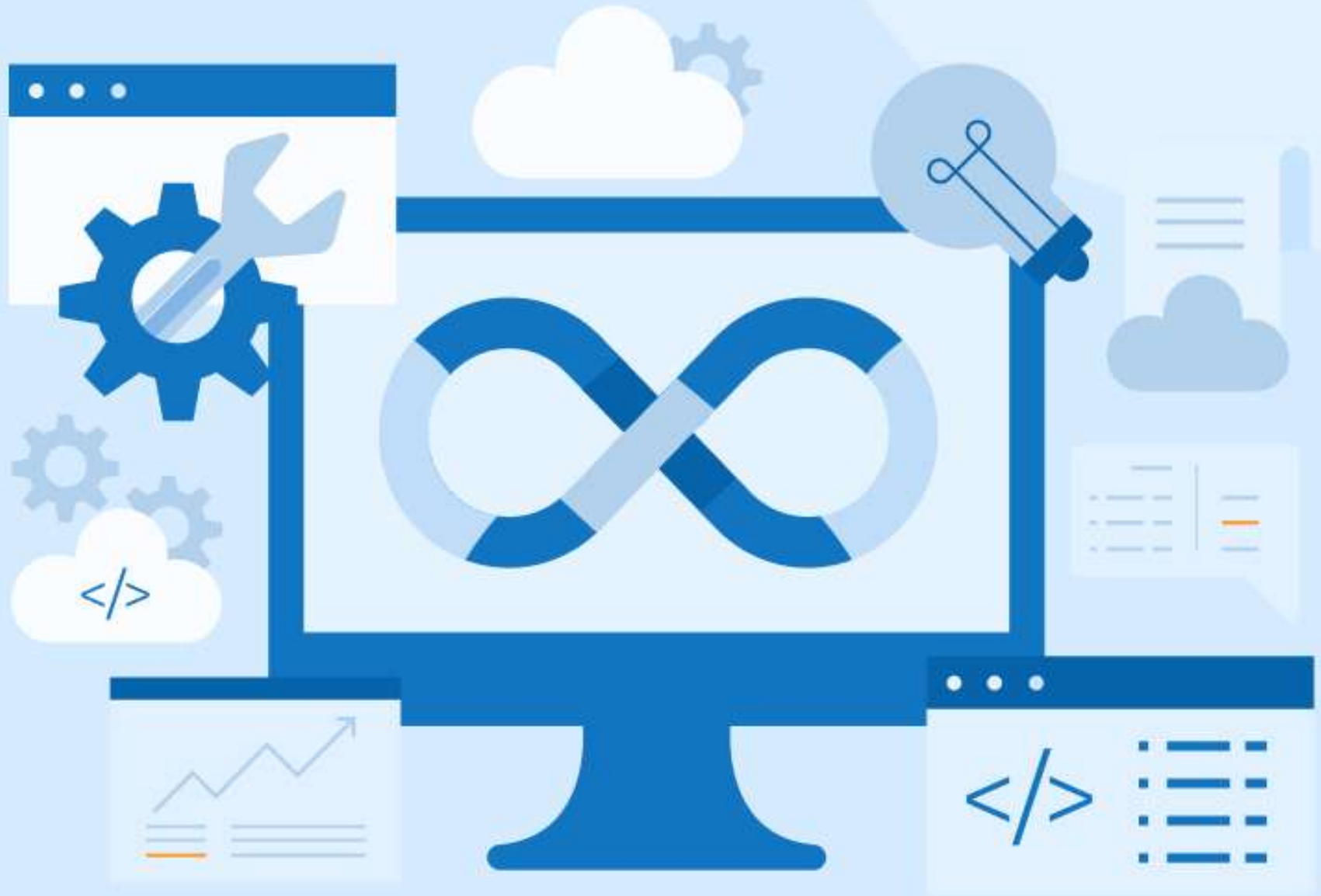
- Build the artifact once to ensure consistent test results.
- Promote the same artifact through all pipeline stages for reliable validation.
- Use a system-agnostic build to separate configurations for consistent testing.

Automate the Tests

- CI/CD includes testing to reduce bugs, not just deployment.
- Automate unit, integration, and functional tests for thorough verification.
- Careful test writing prevents unexpected function changes.

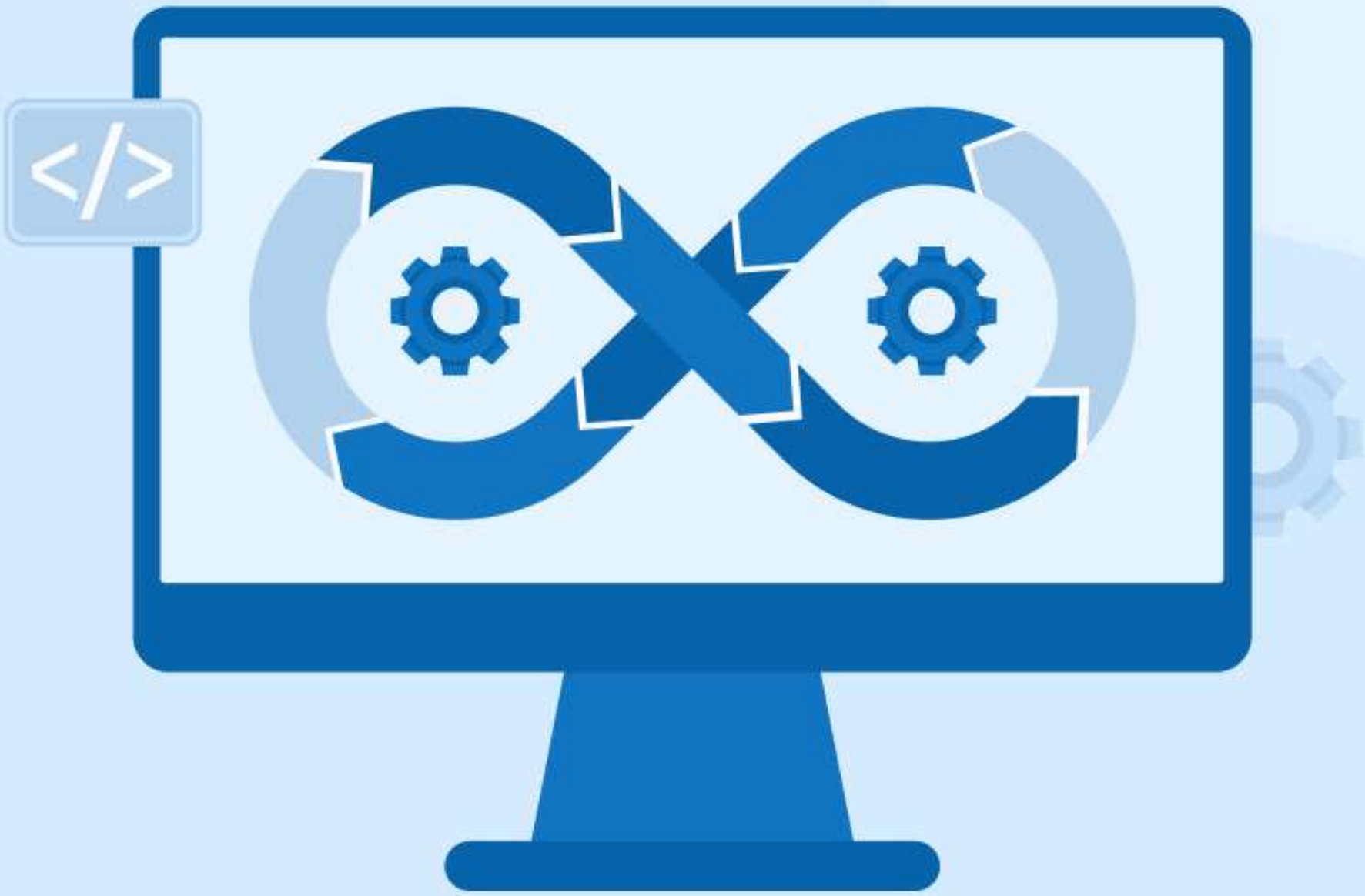
Monitor CI/CD Pipeline

- Small delays can accumulate into major issues in CI/CD pipelines.
- Observability tools offer insights on tests, QA, deployment speed, and builds.
- Continuous monitoring identifies failures and critical areas for improvement.



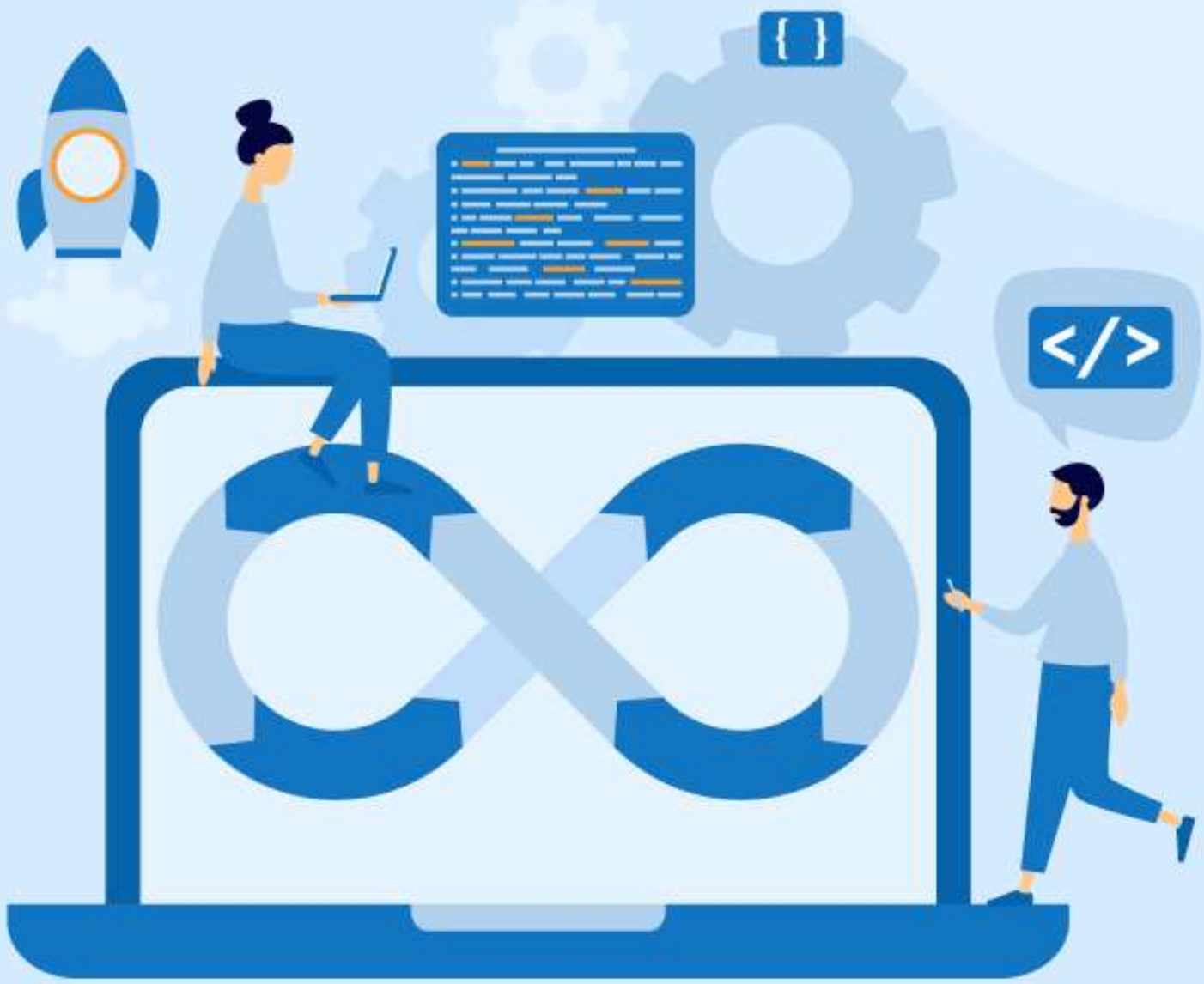
Select Appropriate Tools

- Choose DevOps tools based on project requirements and priorities.
- Prioritize features like security and collaboration when selecting tools.
- Ensure tools provide visibility throughout the CI/CD pipeline.



Make CI/CD Fast and Easy

- Fast CI/CD enables quick developer feedback.
- Slow pipelines increase errors and bugs from workarounds.
- Simplify processes and use caching for efficiency.



Release Often

- Frequent releases allow software upgrades but risk bugs.
- Slow pipelines increase errors and bugs from workarounds.
- Simplify processes and use caching for efficiency.

On-Demand Testing Environments

- Use clean, temporary testing environments in containers for consistent test execution.
- Containers eliminate residual effects, ensuring accurate test results.
- Easy simulation of pipeline configurations and control over local test environments.

Have more insights? Leave your best practices in the comments below!

TatvaSoft
sculpting thoughts...