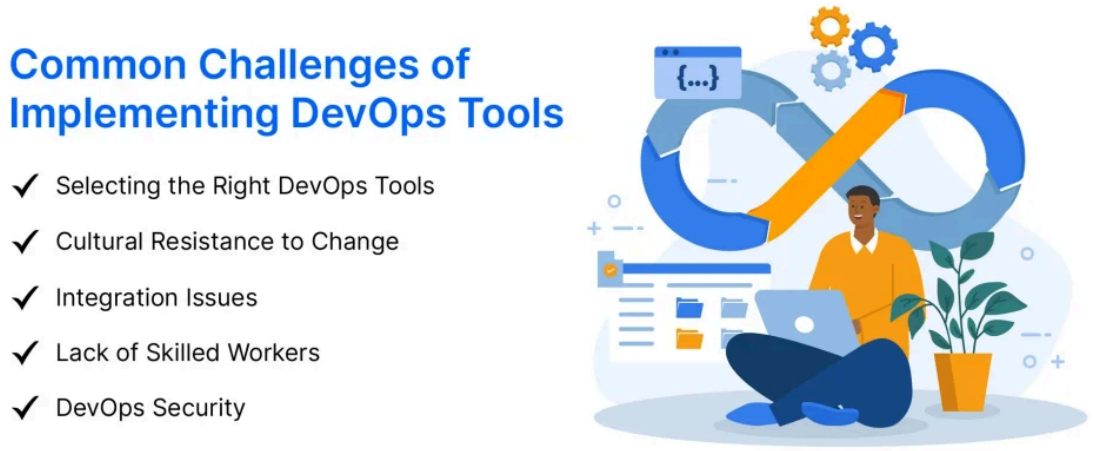
Common DevOps Troubleshooting Scenarios and Solutions

Author: Zayan Ahmed | **Estimated Reading time:** 5 mins

DevOps engineers often face different kinds of technical issues while managing cloud infrastructure and software deployment. Here are some common problems and their solutions explained simply.



1. Application Crashes After Deployment

Scenario: You deploy a new version of your application, but it crashes immediately. The logs show errors, but they are hard to understand.

Solution: First, check the logs carefully using tools like `kubectl logs` (for Kubernetes) or `docker logs` (for Docker). Look for missing dependencies, incorrect environment variables, or database connection issues. If needed, roll back to the previous version and fix the bug before redeploying.

2. High CPU or Memory Usage

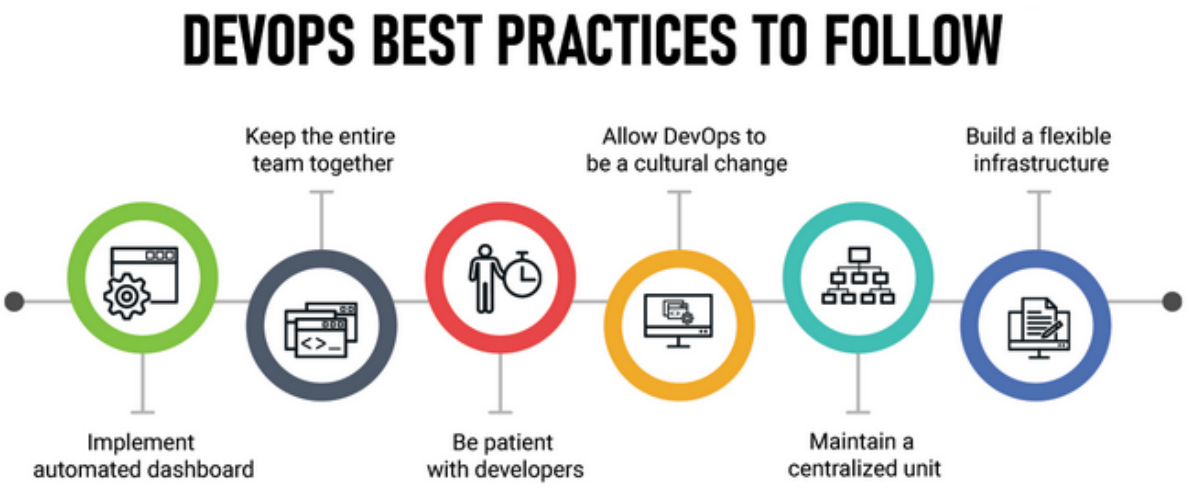
Scenario: Your application is running, but the server is slowing down or even crashing due to high resource usage.

Solution: Use monitoring tools like Prometheus or Grafana to check which process is using too much CPU or memory. If a service is consuming too many resources, consider optimizing the code, adding autoscaling, or increasing instance size.

3. Deployment Fails in CI/CD Pipeline

Scenario: You push your code to Git, but the CI/CD pipeline fails during testing or deployment.

Solution: Check the pipeline logs to see what failed. It could be a syntax error, missing dependencies, or incorrect configuration. Running the pipeline locally before pushing changes can help catch issues early.



4. Network Connectivity Issues

Scenario: Your app cannot connect to the database, API, or another microservice.

Solution: Use tools like `ping` and `curl` to check connectivity. Verify firewall rules, security groups, and network policies. Also, ensure that DNS settings and service discovery configurations are correct.

5. Container Fails to Start

Scenario: A Docker container fails to start, showing errors related to ports, volumes, or permissions.

Solution: Run `docker logs <container_id>` to see what went wrong. Make sure ports are not already in use, volumes are correctly mounted, and the container has the necessary permissions.

6. Slow Application Performance

Scenario: Users complain that your application is slow, taking too long to load pages or process requests.

Solution: Use performance monitoring tools like New Relic or Datadog to find bottlenecks. Optimize slow database queries, enable caching, and check if load balancing is working properly.

7. Security Vulnerabilities

Scenario: A security scan finds vulnerabilities in your software or infrastructure.

Solution: Regularly update dependencies, apply security patches, and follow best practices for securing cloud resources. Use tools like OWASP ZAP for application security testing and enable role-based access control (RBAC) to restrict access.

8. Data Loss or Corruption

Scenario: Important data is lost or gets corrupted due to system failure or accidental deletion.

Solution: Always have backups in place using automated backup solutions. Test your backup and recovery process regularly to ensure you can restore data quickly when needed.

9. Configuration Drift

Scenario: Your infrastructure configuration changes over time, leading to unexpected issues.

Solution: Use Infrastructure as Code (IaC) tools like Terraform or Ansible to maintain consistent configurations. Store all configurations in version control to track and revert changes when necessary.

10. DNS Issues

Scenario: Your domain is not resolving correctly, causing downtime for your application.

Solution: Check your DNS settings, propagation status, and whether your domain is properly linked to the correct IP address or load balancer. Use `nslookup` or `dig` commands to troubleshoot DNS problems.

Conclusion

DevOps troubleshooting requires a mix of monitoring, logging, and systematic problem-solving. By understanding these common issues and solutions, DevOps engineers can keep systems running smoothly and avoid major downtime.