Day-to-Day Tasks of a DevOps Engineer

Written by Zayan Ahmed | 5 min read

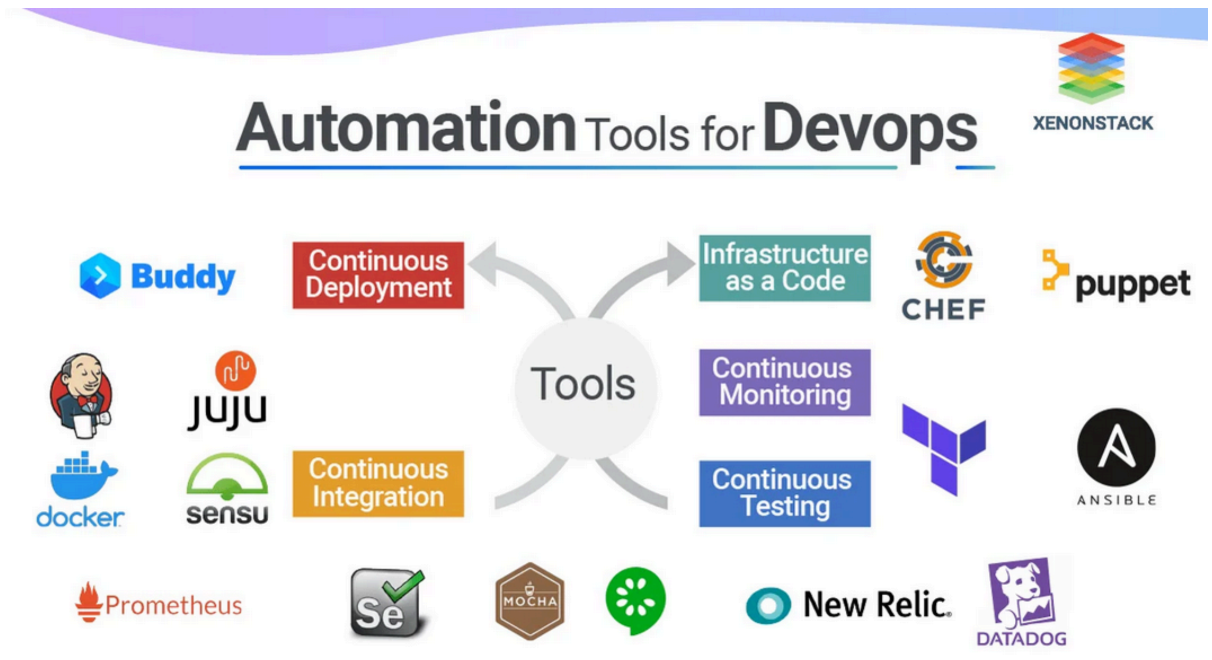
1. Infrastructure Management & Monitoring

- **Check and Monitor Infrastructure**
  First thing in the morning, ensure that all systems and services are operational. Utilise tools like **Prometheus**, **Grafana**, **Datadog**, or **CloudWatch** for health checks on servers, applications, and databases.
  Actions:
  - Monitor uptime, CPU, memory, and network traffic.
  - Review alerts or incidents raised by monitoring systems.
  - Respond to any critical infrastructure issues.
- **Provision New Infrastructure**
  Use Infrastructure as Code (IaC) tools like **Terraform**, **Ansible**, or **CloudFormation** to provision new servers, databases, or networking components.
  Actions:
  - Review requirements for infrastructure changes.
  - Apply code changes and monitor the deployment.
  - Maintain version control for infrastructure code.

2. CI/CD Pipeline Management

- **Monitor and Manage CI/CD Pipelines**
  Ensure that Continuous Integration and Continuous Deployment pipelines (using **Jenkins**, **GitLab CI**, **CircleCI**, etc.) are running smoothly.
  Actions:

    - Investigate any failed builds.
    - Debug issues in the pipeline or testing environment.
    - Make necessary pipeline configuration changes (e.g., update scripts, dependencies).
- **Optimize CI/CD Pipelines**
  Regularly audit pipelines for bottlenecks and optimize them for faster build, test, and deployment times.
  Actions:
    - Reduce redundant steps.
    - Implement caching mechanisms.
    - Parallelize jobs where possible.



3. Automation & Scripting

- **Automate Repetitive Tasks**
  Write or update scripts (Bash, Python, etc.) to automate tasks such as infrastructure provisioning, database backups, or log rotation.
  Actions:
    - Review current manual tasks that can be automated.
    - Write scripts to handle these tasks.
    - Document the automation processes for future reference.
- **Configuration Management**
  Manage and apply configuration changes across environments using tools like **Ansible**, **Chef**, or **Puppet**.
  Actions:
    - Push configuration changes to the desired environment.
    - Ensure that the configuration is applied consistently.

4. Incident Management & Troubleshooting

- **Respond to Alerts and Incidents**
  If any system or service goes down or an issue is raised, investigate the root cause and implement fixes. Use tools like **Splunk**, **ELK (Elasticsearch, Logstash, Kibana)**, or **Sentry** for log analysis and debugging.
  Actions:
  - Review logs and metrics to identify the issue.
  - Work with developers or network teams for resolution.
  - Implement long-term solutions to prevent recurrence.
- **Incident Postmortems**
  After resolving incidents, conduct postmortems to identify what went wrong and how it can be avoided in the future.
  Actions:
  - Document the incident, cause, and resolution.
  - Implement monitoring or automation to prevent similar issues.

5. Security Management

- **Manage Security & Compliance**
  Ensure that infrastructure and applications comply with security standards, regularly audit for vulnerabilities, and implement patches or updates as required.
  Actions:
  - Run security scans (using **Aqua**, **Anchore**, **Clair**).
  - Apply necessary patches to infrastructure or containers.
  - Monitor for security breaches or suspicious activity.
- **Access and Permissions Management**
  Regularly audit and manage access controls, ensuring that only authorized personnel have access to sensitive systems.
  Actions:
  - Review permissions and adjust user roles.
  - Apply the principle of least privilege to critical systems.

6. Cloud and Container Management

- **Manage Cloud Infrastructure (AWS/Azure/GCP)**
  Handle day-to-day operations on the cloud platform, including setting up new services, scaling infrastructure, and cost optimization.
  Actions:
  - Deploy services such as EC2 instances, RDS databases, or S3 buckets.
  - Monitor costs and optimize resources (e.g., right-sizing instances).
  - Implement best practices for cloud architecture (e.g., high availability).
- **Manage Containers and Orchestration (Docker/Kubernetes)**
  Ensure that containerized applications are running efficiently. Manage container orchestration platforms like **Kubernetes** or **Docker Swarm** for scaling, load balancing, and ensuring high availability.
  Actions:
  - Deploy and monitor containers.
  - Manage Kubernetes clusters (scaling, networking, and upgrades).
  - Troubleshoot any issues with containers or pods.

7. Collaboration and Communication

- **Collaborate with Development Teams**
  Work closely with development teams to ensure smooth deployments, handle environment issues, and support ongoing projects.
  Actions:
  - Participate in standups or sprint planning sessions.
  - Assist developers in resolving environment-related issues.
  - Guide the team in best practices for development pipelines.
- **Documentation**
  Maintain comprehensive documentation for infrastructure setups, CI/CD pipelines, and troubleshooting guides for other team members.
  Actions:
  - Update internal wikis or repositories with any changes.
  - Ensure that new team members can easily follow existing processes.

8. Continuous Improvement & Learning

- **Stay Updated with New Tools & Technologies**
  Continuously learn about new tools, technologies, and best practices in DevOps. Attend webinars, read documentation, and test new tools.
  Actions:
  - Explore improvements in existing workflows (e.g., GitOps, advanced Kubernetes techniques).
  - Implement small proof of concepts (POCs) to test new technologies.
  - Attend meetups or industry conferences (virtual or in-person).

9. Backup and Recovery

- **Ensure Regular Backups**
  Set up automated backups for databases, storage, and infrastructure configurations. Ensure that backup policies are well-defined and tested regularly.
  Actions:
  - Monitor backup jobs and address failures.
  - Run recovery tests to verify that backups are usable.
  - Update backup scripts or tools as required.

10. Performance Tuning & Optimization

- **Optimize System and Application Performance**
  Regularly tune the performance of applications, databases, and infrastructure. Ensure systems are optimized for load and resource consumption.
  Actions:
  - Monitor performance metrics.
  - Identify bottlenecks in code, infrastructure, or databases.
  - Make adjustments to improve efficiency (e.g., scaling resources, changing configuration).

Tools and Technologies Often Used

- **Cloud Providers:** AWS, Azure, Google Cloud Platform
- **CI/CD Tools:** Jenkins, GitLab CI, CircleCI
- **Containerization:** Docker, Kubernetes, Docker Swarm
- **IaC:** Terraform, Ansible, CloudFormation
- **Monitoring & Logging:** Prometheus, Grafana, Datadog, ELK Stack, CloudWatch
- **Scripting Languages:** Bash, Python, PowerShell
- **Version Control:** Git, GitHub, Bitbucket

Follow me on **LinkedIn** for more 😊