**DevOps** training programs are designed to equip individuals and teams with the knowledge and skills necessary to implement and manage DevOps practices effectively. Here’s a comprehensive list of topics typically covered in DevOps training:

1. \*\*Introduction to DevOps\*\*

- \*\*Overview of DevOps\*\*: History, principles, and benefits.

- \*\*DevOps Culture\*\*: Collaboration, communication, and breaking down silos.

- \*\*DevOps Practices and Terminology\*\*: Key concepts and practices.

2. \*\*Version Control Systems\*\*

- \*\*Git Basics\*\*: Repository creation, commits, branches, merges.

- \*\*Advanced Git\*\*: Rebasing, resolving conflicts, tag management.

- \*\*Version Control Platforms\*\*: GitHub, GitLab, Bitbucket basics and features.

3. \*\*Continuous Integration (CI)\*\*

- \*\*CI Concepts\*\*: Understanding CI and its benefits.

- \*\*CI Tools\*\*: Jenkins, Travis CI, CircleCI, GitLab CI/CD.

- \*\*Pipeline Configuration\*\*: Building and managing CI pipelines.

4. \*\*Continuous Delivery and Deployment (CD)\*\*

- \*\*CD Concepts\*\*: Differences between continuous delivery and continuous deployment.

- \*\*Deployment Strategies\*\*: Blue/green deployments, canary releases, rolling updates.

- \*\*CD Tools\*\*: Implementing deployment pipelines with tools like Jenkins, GitLab, or Spinnaker.

5. \*\*Infrastructure as Code (IaC)\*\*

- \*\*IaC Basics\*\*: Definition, benefits, and use cases.

- \*\*IaC Tools\*\*: Terraform, AWS CloudFormation, Ansible.

- \*\*IaC Practices\*\*: Writing and managing IaC scripts, modularization, and versioning.

6. \*\*Configuration Management\*\*

- \*\*CM Concepts\*\*: Overview and importance.

- \*\*Configuration Management Tools\*\*: Puppet, Chef, Ansible basics.

- \*\*Managing Configurations\*\*: Automation of server configurations and updates.

7. \*\*Containerization\*\*

- \*\*Docker Fundamentals\*\*: Docker images, containers, Dockerfile, Docker Compose.

- \*\*Container Orchestration\*\*: Kubernetes basics, managing clusters, deploying applications.

8. \*\*Cloud Computing\*\*

- \*\*Cloud Basics\*\*: Introduction to cloud services and models (IaaS, PaaS, SaaS).

- \*\*Cloud Providers\*\*: AWS, Azure, Google Cloud Platform.

- \*\*Key Services\*\*: Compute, storage, databases, networking, serverless.

9. \*\*Monitoring and Logging\*\*

- \*\*Monitoring Tools\*\*: Prometheus, Grafana, Nagios, New Relic basics.

- \*\*Logging Tools\*\*: ELK Stack (Elasticsearch, Logstash, Kibana), Splunk.

- \*\*Setting Up Monitoring and Alerts\*\*: Configuring alerts, visualizing metrics.

10. \*\*Security in DevOps (DevSecOps)\*\*

- \*\*Security Principles\*\*: Integrating security into the DevOps pipeline.

- \*\*Security Tools\*\*: Vulnerability scanning, static code analysis, compliance tools.

- \*\*Best Practices\*\*: Implementing security policies and practices in CI/CD.

11. \*\*Collaboration and Communication\*\*

- \*\*Tools\*\*: Slack, Microsoft Teams, JIRA, Confluence.

- \*\*Agile Methodologies\*\*: Scrum, Kanban basics.

- \*\*Effective Communication\*\*: Practices for improving team collaboration and transparency.

12. \*\*Performance Optimization\*\*

- \*\*Performance Monitoring\*\*: Application Performance Monitoring (APM) tools.

- \*\*Load Testing\*\*: Techniques and tools for stress and performance testing.

13. \*\*Disaster Recovery and High Availability\*\*

- \*\*Backup Strategies\*\*: Automated backups, redundancy, failover mechanisms.

- \*\*Disaster Recovery Planning\*\*: Creating and testing disaster recovery plans.

14. \*\*Release Management\*\*

- \*\*Release Strategies\*\*: Feature flags, phased rollouts, blue/green deployments.

- \*\*Change Management\*\*: Processes for managing and rolling out changes.

15. \*\*Metrics and Reporting\*\*

- \*\*Key Metrics\*\*: Deployment frequency, lead time for changes, mean time to recovery (MTTR).

- \*\*Reporting Tools\*\*: Generating and analyzing reports to track and improve DevOps practices.

16. \*\*Automation and Scripting\*\*

- \*\*Scripting Basics\*\*: Bash, Python, PowerShell.

- \*\*Automation Tools\*\*: Automating repetitive tasks and processes.

17. \*\*Soft Skills and Best Practices\*\*

- \*\*Problem-Solving Skills\*\*: Troubleshooting and debugging.

- \*\*Best Practices\*\*: Adopting best practices for DevOps implementation and continuous improvement.

18. \*\*Hands-On Labs and Projects\*\*

- \*\*Practical Exercises\*\*: Building CI/CD pipelines, deploying applications, managing containers.

- \*\*Case Studies\*\*: Real-world scenarios and solutions.

**DevSecOps - Integrating Security in DevOps.**

**Step1-build node (required s/w installed over rehydration) with below prerequisites**

install java 11

build node setup and automate to start.

to build code-install prerequisites softwares-maven, nodejs ant.php, Fortify etc.,

**Step2-create jenkins pipeline with stages.**

specify the buildnode/agent lebel/Jenkins slave

**stage1**-Checkout code from SCM

**stage2** build process

**stage3**-Security Scans

**stage1-Fortify** based on project name,evn, /apps/opt/mw/scripts/fortifyscan.sh

**stage2 -Blackduck** download the bd build cmd script, run from workspace

**stage3 -sonarqube** with build breaker(sonar fails/does not meet, it won't move further Dsonar buildbreaker skip=false) https://onesonarcloud venzon.com/projects

**stae4** -Deployment

**stage1** -upload artifactory

**stage2**-deploy servers in the respective environments

**stage5** -unit test 'place Holders' maintained.

**Post** -success/failure post notifications to who executes the job.

Stop gate expansion - signature scan

**Fortify** is primarily focused on identifying vulnerabilities within the application's source code,

while **Black Duck** specializes in tracking open-source components and managing the associated security risks. Depending on your specific needs and the nature of your application, you might use one or both tools in your software development and security processes.

**SonarQube:** Quality Assurance, continuous code analysis with SonarQube helps maintain high code quality.

**A DAST scan**, or Dynamic Application Security Testing, is a security testing technique that examines an application while it's running to identify vulnerabilities and weaknesses. It's often used to assess web applications for common security issues like SQL injection, cross-site scripting (XSS), and insecure authentication mechanisms.