Dev-Ops Roadmap

Start Here

Hey friend! Looks like you're here because you're interested in learning more about the Dev-Ops roadmap. Awesome! Let's start with a few things first.

Understanding your current background and your goals will help you better navigate the roadmap. Here's a few questions you should think about before you start:

- Is you're a software developer?
- Do you want to be a Dev-Ops Engineer?
- Are you a student looking for a career in Dev-Ops?
- Are you looking to move into a Dev-Ops Technology?

Now that I know what you are, let's talk about what Dev-Ops is and what it means to you.

What is Dev-Ops?

Dev-Ops is a set of practices that combine software development (Dev) and IT operations (Ops) with the goal of shortening the system development life cycle while providing continuous delivery with high software quality.

Why Dev-Ops is important?

Dev-Ops is important because it allows companies to deliver their products faster and more reliably, which can give them a competitive advantage in the market. Dev-Ops also helps to improve collaboration between different teams, such as developers and IT operations, which can lead to more efficient and effective software development and deployment processes.

What does a Dev-Ops Engineer do?

A Dev-Ops Engineer is responsible for implementing and managing the tools and processes that support the Dev-Ops practices. This includes automating the software development and deployment process, monitoring the performance of the software and infrastructure, and ensuring the security and compliance of the software and infrastructure.

Dev-Ops Engineer Skills

Some of the skills that are important for a Dev-Ops Engineer to have include:

- Knowledge of software development and IT operations
- Proficiency in scripting and automation
- Experience with cloud computing and containerization
- Understanding of security and compliance
- Strong communication and collaboration skills

1.Concepts of Software Development

Understanding the concepts of software development is important for a Dev-Ops Engineer as it allows them to understand the software development process and how it can be optimized for faster and more reliable delivery.

Some popular concepts include the waterfall model, agile methodology, lean, kanban, scrum, extreme programming, feature driven development, dynamic systems development method, crystal, and rational unified process.

Each concepts of learning depends on a deeper understanding.

Version Control, Testing, Deployment, Monitoring, Feedback

Agile Methodology, Waterfall Model

2. OS & Linux Basics

As a Dev-Ops Engineer you are responsible for automating and maintaining the infrastructure that supports the software development process.

Linux is a popular open-source operating system that is commonly used in Dev-Ops environments. It provides a stable and secure platform for running applications and services.

Some concepts of learning Systems are:

- Linux Commands
- Linux Administration
- Linux Networking
- Linux Security
- Shell Scripting
- Kernel Internals and Kernel Modules
- Linux Performance Tuning
- Linux Virtualization and Containerization (Docker, Kubernetes, LXC/LXD)
- Networking Protocols (TCP/IP, HTTP, DNS, DHCP)
- Network Configuration and Troubleshooting
- Firewalls and Network Security
- Backup and Disaster Recovery
- Scripting and Automation (Bash, Python, Perl)
- Performance Monitoring and Optimization
- Troubleshooting and Debugging
- Cloud Computing (AWS, Azure, GCP)
- Containerization (Docker, Kubernetes)
- Configuration Management (Ansible, Puppet, Chef)
- Continuous Integration and Continuous Deployment (CI/CD)
- Monitoring and Logging (Prometheus, Grafana, ELK Stack)
- Security and Compliance (SELinux, AppArmor, PCI DSS, HIPAA)
- Performance Tuning and Optimization
- Troubleshooting and Debugging
- Cloud Computing (AWS, Azure, GCP)
- Containerization (Docker, Kubernetes)
- Configuration Management (Ansible, Puppet, Chef)
- Continuous Integration and Continuous Deployment (CI/CD)
- Monitoring and Logging (Prometheus, Grafana, ELK Stack)
- Security and Compliance (SELinux, AppArmor, PCI DSS, HIPAA)
- Performance Tuning and Optimization
- Troubleshooting and Debugging

3. Containerization - Docker

As a Dev-Ops Engineer, Docker is a popular tool for containerization and virtualization. It allows you to package your applications and their dependencies into a lightweight, portable container that can be easily deployed across different environments.

Some Docker concepts to learn include:

- Docker Basics
- Docker Image Management
- Docker Networking
- Docker Volumes
- Docker Security
- Docker Compose
- Docker Swarm
- Docker Performance Tuning
- Docker Troubleshooting and Debugging

4. CI/CD Pipelines

As a Dev-Ops Engineer, understanding CI/CD pipelines is important for automating the software development process and ensuring the delivery of high-quality software.

Some CI/CD concepts to learn include:

- Version Control
- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Pipeline Stages
- Pipeline Triggers
- Artifacts Management
- Testing in the pipeline

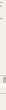
5. Learn one Cloud Provider

Learning a cloud provider is important for a Dev-Ops Engineer as it allows them to leverage the cloud infrastructure and services to automate the deployment and management of applications and infrastructure. Learning a single cloud provider also helps to deepen the understanding of cloud concepts and services, which can be applied to other cloud providers.

Learn one cloud provider and stick with it. Learn the basics and understand how they work. Don't worry about learning all three cloud providers at once.

6. Container Orchestration - Kubernetes

As a Dev-Ops Engineer, container orchestration is important for managing and scaling containerized applications in a production environment. Kubernetes is a popular open-source platform for container orchestration that provides a wide range of features and capabilities for managing containerized applications.



Some Kubernetes concepts to learn include:

- Kubernetes Architecture
- Kubernetes Objects
- Kubernetes Networking
- Kubernetes Storage
- Kubernetes Security
- Kubernetes Scaling and Load Balancing
- Kubernetes Monitoring and Logging
- Kubernetes Troubleshooting and Debugging

7. Monitoring & Observability

Often problems in production. It's important to monitor it in order to find performance issues and troubleshoot problems before they impact your customers.

Some Monitoring and Observability concepts you should learn include:

- Monitoring Fundamentals
- Metrics Collection and Aggregation
- Logs Management and Analysis
- Distributed Tracing
- Alerting and Notification
- Dashboards and Visualization
- Performance Monitoring and Optimization
- Troubleshooting and Debugging

8. Infrastructure as Code

Dev-Ops is a key concept in DevOps that emphasizes the use of code to define and manage infrastructure rather than manual processes or configurations. This approach allows for automation, consistency, and scalability in managing infrastructure.

Some Infrastructure as Code concepts to learn include:

- Infrastructure as Code Principles
- Configuration Management
- Infrastructure Provisioning
- Version Control for Infrastructure
- Automated Testing for Infrastructure
- Security and Compliance in Infrastructure as Code
- Monitoring and Logging in Infrastructure as Code
- Scaling and Load Balancing in Infrastructure as Code
- Disaster Recovery in Infrastructure as Code
- Troubleshooting and Debugging in Infrastructure as Code

9. Scripting Language

Scripting languages are important for automating repetitive and time-consuming tasks in software development and IT operations. They provide a flexible and lightweight way to automate these tasks and can be used to manage and orchestrate complex workflows.

Some popular scripting languages to learn include:

- Bash
- Python
- Perl
- Ruby
- PowerShell
- JavaScript (Node.js)
- Groovy
- Lua
- Tcl/Tk
- PHP

Some topics to be familiar with when learning a scripting language include:

- Syntax and Data Structures
- Control Flow and Loops
- Functions and Modules
- File Handling and Input/Output
- Regular Expressions
- Error Handling and Debugging
- Automation and Scripting for Common Tasks
- Integration with Other Tools and Systems
- Performance Optimization
- Security and Compliance in Scripting

10. Version Control - Git

Version control systems are important for software development teams to manage and track changes to the source code and collaborate on the development process.

Git is one of the most popular version control systems and is widely used in the industry. Some of the key concepts to learn in Git include:

- Repository Management
- Branching and Merging
- Commit Management
- Remote Repositories
- Collaboration and Pull Requests
- Conflict Resolution
- Git Hooks
- Git Workflows
- Advanced Git Commands
- Security and Access Control
- Performance Optimization
- Troubleshooting and Debugging

Starting from.

Starting from Scratch offers a lot of flexibility but also requires a deep understanding of the underlying technologies and processes. It can be a challenging but rewarding approach for those who have the time and expertise to build a robust and customized Dev-Ops pipeline from scratch.

Starting from a Cloud Provider's Managed Services can provide a faster and more streamlined approach to Dev-Ops, but it may limit the flexibility and control over the underlying infrastructure and services.

Starting from an Existing Framework or Platform can provide a good balance between flexibility and ease of use, but it may require some customization and integration with other tools and services.

Starting as a... Systems Administrator

Systems Administrators are responsible for the maintenance and upkeep of computer systems and networks. They are responsible for ensuring the stability and security of these systems, including installing and configuring software, monitoring performance, and troubleshooting issues.

Some skills that are important for a Systems Administrator include:

- Operating Systems: Linux, Windows, macOS
- Networking: TCP/IP, DNS, DHCP, VPN
- Security: Firewalls, Intrusion Detection and Prevention Systems, Encryption
- Scripting: Bash, Python, PowerShell
- Virtualization: VMware, Hyper-V, KVM
- Cloud Computing: AWS, Azure, GCP
- Monitoring: Nagios, Zabbix, Prometheus
- Backup and Recovery: Tape, Disk, Cloud
- Troubleshooting: Diagnosing and resolving issues in a timely manner

Starting as a... Test Automation Engineer

Test Automation Engineers are responsible for creating and maintaining automated tests to ensure the quality and reliability of software applications. They work closely with developers and testers to identify and prioritize test cases, write test scripts, and integrate them into the software development and testing pipeline.

Some skills that are important for a Test Automation Engineer include:

- Programming: Java, Python, JavaScript, C#
- Test Automation Frameworks: Selenium, Cypress, Appium, JUnit, TestNG
- Version Control: Git, SVN
- Continuous Integration and Continuous Deployment: Jenkins, GitLab CI/CD, CircleCI
- Cloud Computing: AWS, Azure, GCP
- Containerization: Docker, Kubernetes
- Performance Testing: JMeter, Gatling, Locust
- Security Testing: OWASP ZAP, Burp Suite
- Mobile Testing: Appium, Espresso, XCUITest
- Web Testing: Selenium, Cypress, Puppeteer
- Test Data Management: Test data generation, masking, and management
- Test Reporting and Analytics: Allure, ReportPortal, Elasticsearch, Kibana
- Troubleshooting: Diagnosing and resolving issues in a timely manner

Starting as a... Network Engineer

Network Engineers are responsible for the design, implementation, and maintenance of computer networks. They are responsible for ensuring the stability and security of these networks, troubleshooting issues, and optimizing performance.

Some skills that are important for a Network Engineer include:

- Networking Fundamentals: TCP/IP, DNS, DHCP, VPN
- Network Hardware: Routers, Switches, Firewalls, Load Balancers
- Network Security: Firewalls, Intrusion Detection and Prevention Systems, Encryption
- Cloud Networking: AWS, Azure, GCP
- Virtualization: VMware, Hyper-V, KVM
- Scripting: Bash, Python, PowerShell
- Monitoring: Nagios, Zabbix, Prometheus
- Troubleshooting: Diagnosing and resolving issues in a timely manner
- Performance Optimization: Optimizing network performance and bandwidth usage
- Disaster Recovery: Backup and recovery of network data and configurations
- Documentation: Creating and maintaining network diagrams and documentation
- Collaboration: Working with other teams and stakeholders to design and implement network solutions

Starting with... No or little IT Background

Many people start their IT journey from scratch, with little or no prior experience in the field. This can be a challenging but rewarding path, as it allows individuals to build a strong foundation in the skills and knowledge required for a successful career in IT.

Some of the skills that are important for a beginner in IT include:

- Basic computer literacy: understanding how to use a computer, navigate the operating system, and install and manage software.
- Problem-solving: the ability to think critically and logically to identify and resolve issues and challenges.
- Learning ability: the willingness and ability to learn new skills and keep up with the latest trends and technologies in the field.
- Communication: the ability to communicate effectively with colleagues and stakeholders, both verbally and in writing.
- Teamwork: the ability to work collaboratively with others to achieve common goals.
- Time management: the ability to prioritize and manage tasks and deadlines effectively.
- Attention to detail: the ability to pay close attention to details and ensure accuracy in work.
- Adaptability: the ability to adapt to changes and new challenges in the workplace.

Summary DevOps Roadmap

1. Starting Fundamentals Skill
   - Basic understanding of Software Development Life Cycle (SDLC) and Agile methodologies
   - Basic understanding of Linux/Unix operating systems, including command-line interfaces and basic scripting
   - Basic understanding of networking concepts, including TCP/IP, DNS, HTTP, and firewalls
   - Basic understanding of version control systems, such as Git, and how to create, clone, commit, push, and pull changes to a repository
   - Basic understanding of cloud computing concepts, including IaaS, PaaS, and SaaS, and familiarity with at least one cloud provider (AWS, Azure, or GCP)
   - Basic understanding of containerization and virtualization, including Docker and Kubernetes
   - Basic understanding of continuous integration and continuous deployment (CI/CD) pipelines, including tools like Jenkins, GitLab CI/CD, and CircleCI
   - Basic understanding of monitoring and logging, including tools like Prometheus, Grafana, and ELK Stack
   - Basic understanding of security and compliance, including topics like SSL/TLS, firewalls, and access control
2. Learning Intermediate Skill
   - Advanced understanding of Linux/Unix operating systems, including advanced scripting and troubleshooting
   - Advanced understanding of networking concepts, including topics like load balancing, VPNs, and DNS
   - Advanced understanding of version control systems, including branching and merging strategies, and advanced Git commands
   - Advanced understanding of cloud computing concepts, including topics like serverless computing, containerization, and cloud-native architectures
   - Advanced understanding of containerization and virtualization, including topics like Kubernetes, Docker Swarm, and container security
   - Advanced understanding of CI/CD pipelines, including topics like pipeline as code, automated testing, and deployment strategies
   - Advanced understanding of monitoring and logging, including topics like distributed tracing, log aggregation, and anomaly detection
   - Advanced understanding of security and compliance, including topics like vulnerability scanning, penetration testing, and compliance frameworks
3. Learning Advanced Skill
   - Expertise in designing and implementing complex DevOps pipelines, including multi-cloud and hybrid cloud environments
   - Expertise in automating infrastructure provisioning and configuration management using tools like Terraform, Ansible, and Puppet
   - Expertise in optimizing performance and scalability of applications and infrastructure, including topics like caching, load balancing, and CDNs
   - Expertise in implementing advanced security and compliance measures, including topics like zero trust architecture, identity and access management, and data encryption
   - Expertise in troubleshooting and debugging complex issues across multiple systems and components, including topics like root cause analysis and performance profiling
   - Expertise in leading and mentoring other DevOps engineers and teams, including topics like knowledge sharing, code reviews, and pair programming
   - Expertise in staying up-to-date with the latest trends and technologies in the DevOps ecosystem, including topics like emerging tools and frameworks, and industry best practices.