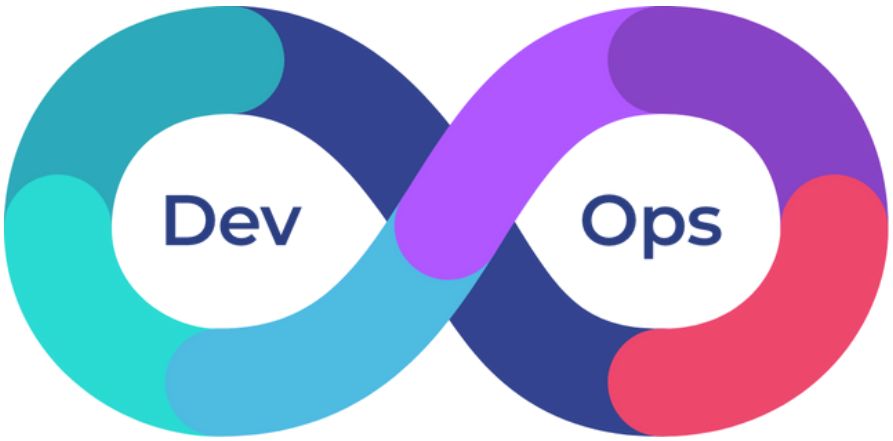
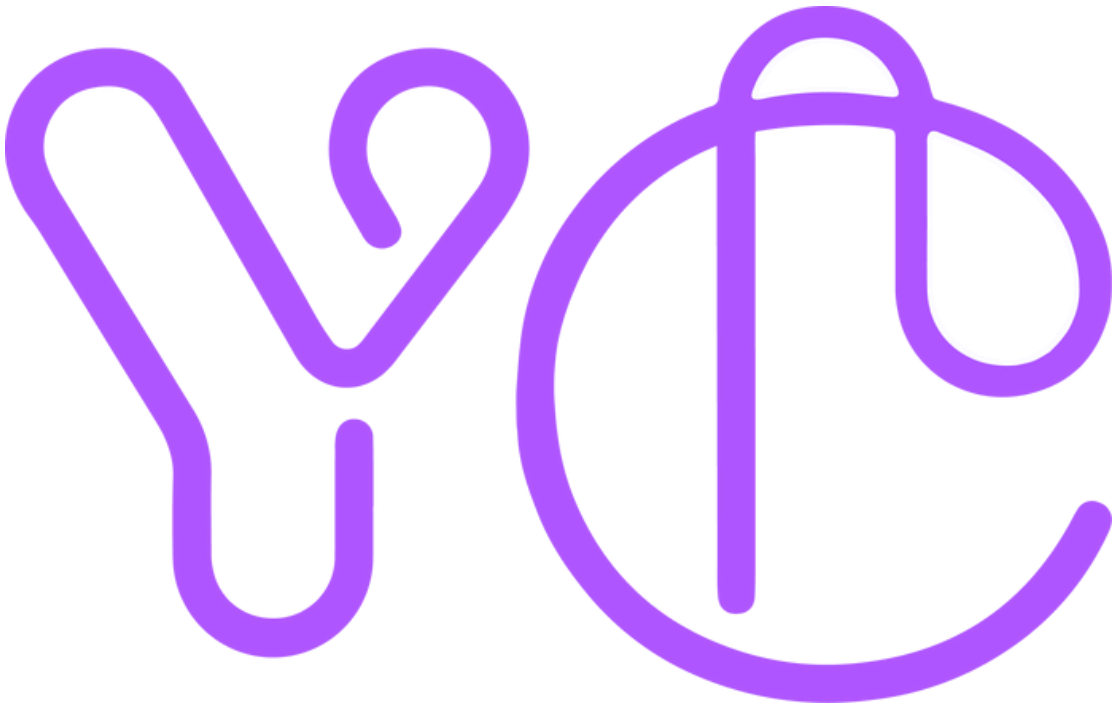
DEVOPS
ROADMAP



Yatharth Chauhan

WELCOME TO MY WORLD



yatharthchauhan.me

CONNECT WITH ME

DevOps Engineer ROADMAP

What is a DevOps Engineer?

DevOps engineers use tools and techniques to automate and improve the process of building, testing, and releasing software. This helps to get new features and updates to users faster and with fewer problems.

A DevOps engineer is like a bridge builder between two teams:

- **Developers**: They create software.
- **Operations**: They make sure the software runs smoothly.

What Does a DevOps Engineer Do?

Here's what a DevOps engineer is responsible for:

- **Building Bridges**: They create tools and processes to make communication between developers and operations teams smoother.
- **Automation**: They use tools to automate tasks like testing and releasing software, making things faster and more efficient.
- **Making Things Run Smoothly**: They ensure that software updates get to users quickly and without causing problems.

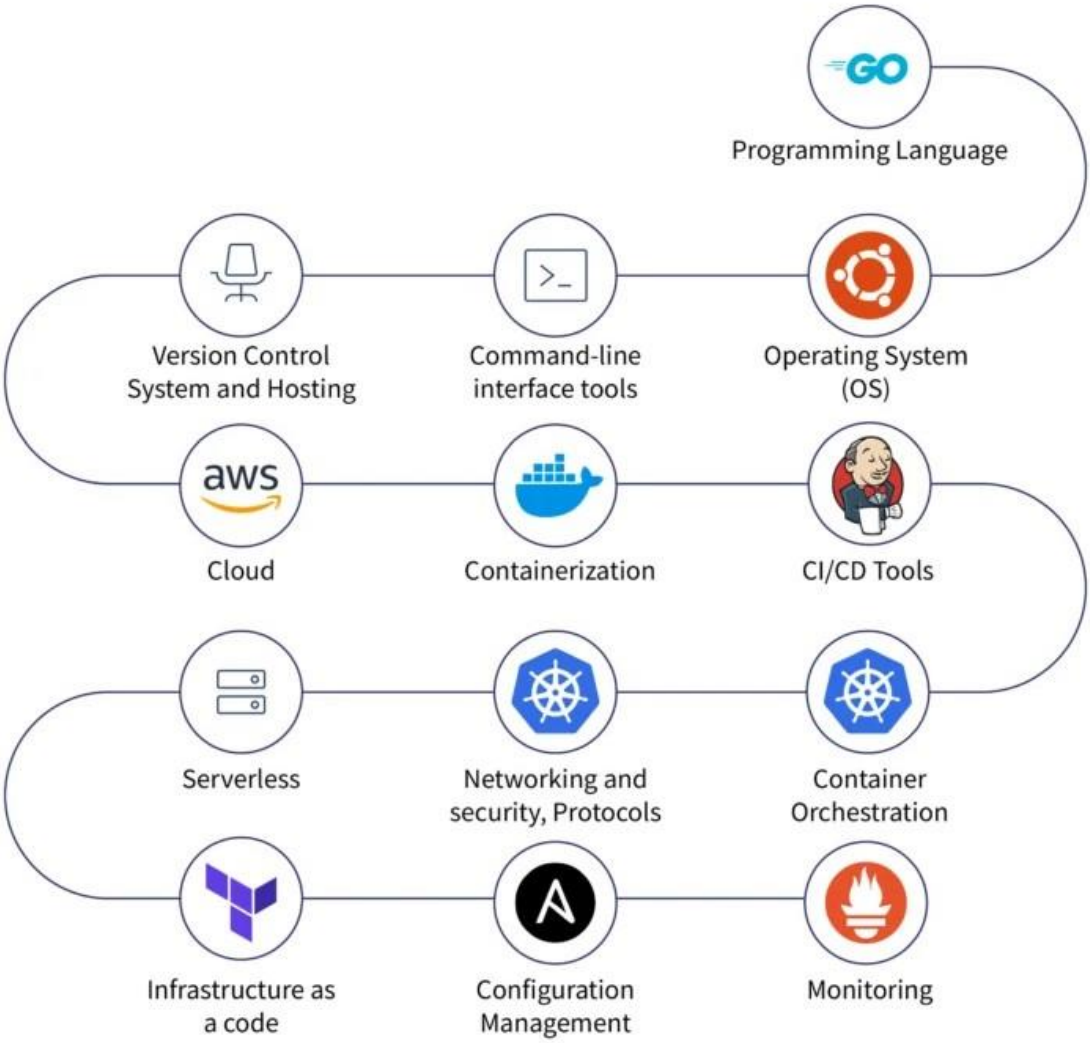
DevOps Engineer Prerequisites and Qualifications

Technical Knowledge

Whether you have a formal degree or are self-taught, focusing on these technical subjects is essential:

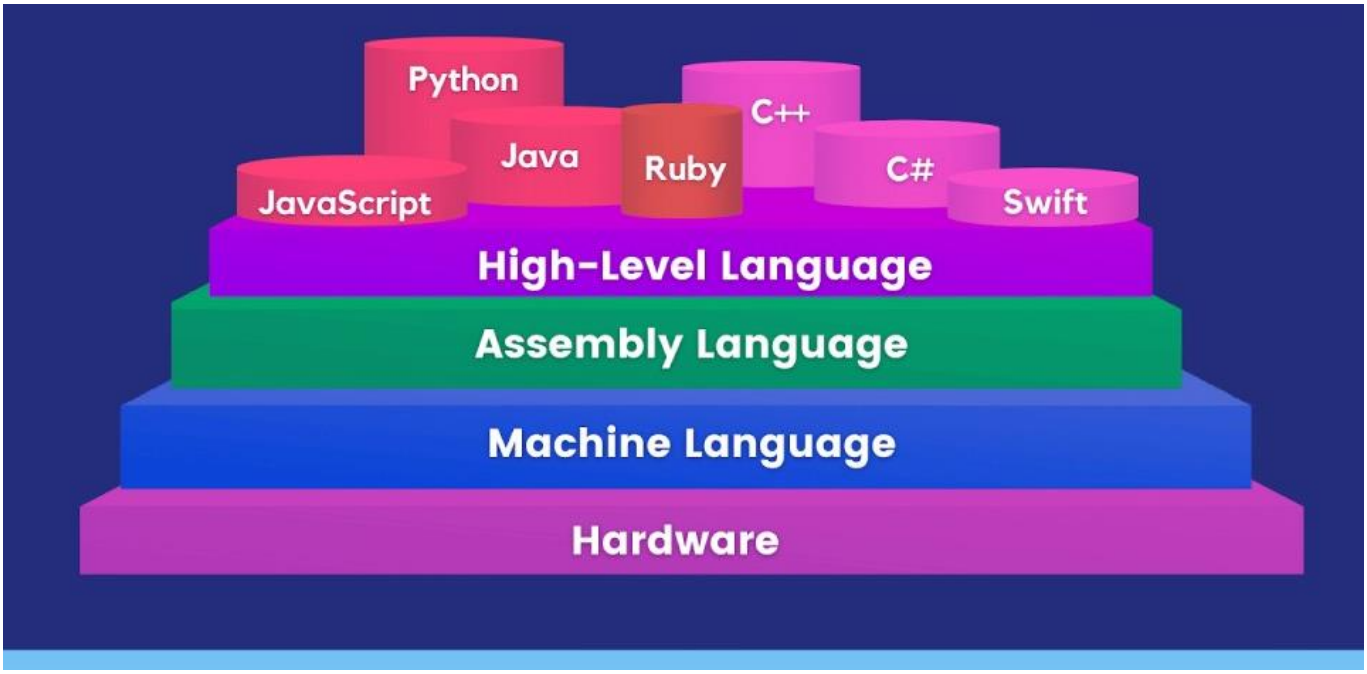
- **Operating System**: Understanding how operating systems work is crucial for managing infrastructure and deploying software.
- **Computer Networks**: Understanding computer networks is important for managing network configurations and troubleshooting connectivity issues.
- **Computer Networks**: Understanding computer networks is important for managing network configurations and troubleshooting connectivity issues.
- **Distributed Systems (Advanced)**: Advanced knowledge of distributed systems is beneficial for handling large-scale applications and cloud computing environments.

DEVOPS ROADMAP



Step 1: Select a Programming Language

Learning a programming language is a valuable skill for DevOps Engineers because you'll use it to automate tasks and solve problems. Even if the project uses a different language, the concepts are similar across languages.



Here are some top programming languages to consider:

- **Python**: Easy to learn and widely used for automation and scripting tasks.
- **Bash**: Built-in to most Unix-like systems, great for shell scripting and automation.
- **JavaScript**: Useful for web development and working with cloud platforms.
- **Go**: Known for its efficiency and suitability for building tools and microservices.
- **Ruby**: Often used with tools like Puppet and Chef for configuration management.

Step 2: Basics of Operating System (OS)

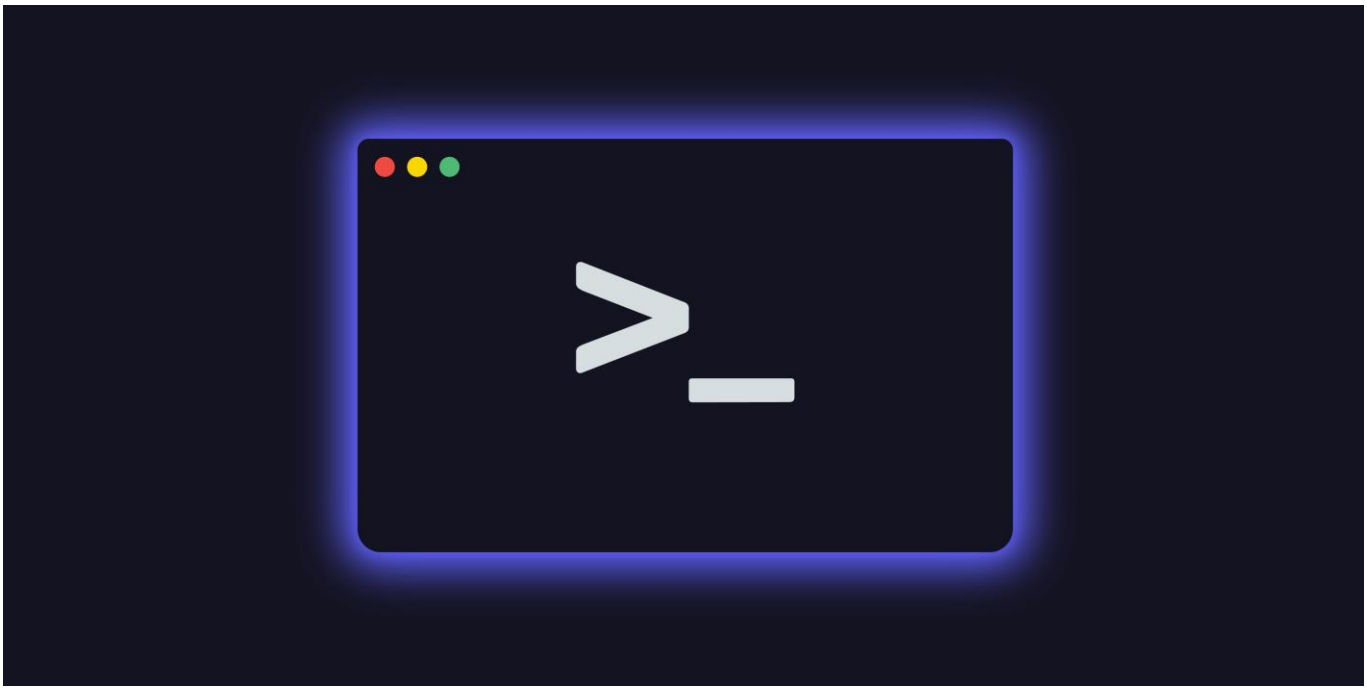
Operating systems are like the managers of computers. They control how software and hardware work together. To be a good DevOps Engineer, you need to understand how operating systems work.

Here are some important ones to start with:

- **Windows**: This is the most common operating system for personal computers. It's used by many companies and individuals around the world.
- **Linux**: Linux is an open-source operating system used in many servers and devices. It's powerful and flexible, making it popular for running websites and applications.
- **Ubuntu / Debian**: These are popular versions of Linux. They are user-friendly and have a lot of community support, making them great choices for beginners.

Step 3: Command-line Interface (CLI)

Understanding the Command Line Interface (CLI) is really important for a DevOps Engineer. It's like having superpowers to control your system or tools directly through text commands instead of clicking around in a graphical interface.



Why CLI Matters?

CLI gives you:

- More advanced features hidden in graphical interfaces.
- Precise control over your system or tools.
- Common functionality across different environments and operating systems.
- Remote access to infrastructure and systems from anywhere.

What to Learn?

To become good at CLI, focus on:

- **Scripting**: Writing scripts to automate tasks.
- **Editors**: Working efficiently with text editors for writing and editing code.
- **Networking Tools**: Understanding tools for managing network connections and configurations.
- **Process Monitoring**: Keeping an eye on running processes and managing them effectively.
- **Performance Monitoring**: Monitoring system performance to identify and fix issues.
- **Text Manipulation**: Learning how to manipulate text files and data using command-line tools.

Step 4: Version Control System and Hosting

Including version control and hosting in your DevOps journey is crucial because it helps with collaboration, managing code, and tracking versions. These are vital for the DevOps lifecycle or if you're aiming for the GitOps approach.

Version Control System – GIT

Git is a widely used distributed version control system. Here's what you need to know:

- **Repositories**: Places where your code is stored.
- **Branching**: Creating separate lines of development.
- **Commits and Merges**: Saving changes and combining them.
- **Tracking Changes**: Monitoring modifications made to code.
- **Collaboration**: Working together on projects.

Hosting (GitHub, Bitbucket, Gitlab)

Once your code is managed using Git, you need to host it so it's accessible. Here are some hosting platforms:

- **GitHub**: Popular for open-source projects and collaboration.
- **Bitbucket**: Often used for private repositories within organizations.
- **Gitlab**: Offers a complete DevOps platform with integrated CI/CD.

Step 5: Learn About Cloud Providers

In the DevOps roadmap, understanding cloud computing is crucial because most applications are hosted either on a cloud or on-premises servers.

Here are some major cloud providers in 2024:

AWS (Amazon Web Services)

- AWS offers a variety of services like EC2 for virtual servers, S3 for storage, Lambda for serverless computing, and EKS for managing Kubernetes clusters. These services are helpful for deploying your software.

Microsoft Azure

- Azure is provided by Microsoft and offers similar services to AWS. One notable aspect for DevOps engineers is the Azure DevOps tools, which include integrated CI/CD tools, version control, and project management tools.

GCP (Google Cloud Platform)

- Google Cloud Platform (GCP) provides services similar to other cloud providers and is offered by Google.

While most cloud providers offer similar services, there are differences in pricing and some unique services specific to each provider. It's important to understand these differences when choosing a cloud provider for your projects.

Step 6: Containerization – Docker

Docker is a super useful tool for DevOps Engineers. Before Docker, deploying an app was a pain. You had to bundle up the app and all its stuff, then install everything it needed on the server. It was a headache, and making it work for a lot of users was even harder.

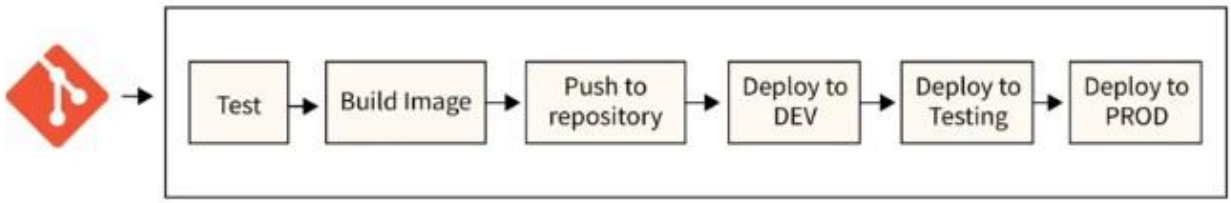
But Docker changes the game. Now, every organization loves Docker because it's so easy. You just write a simple file called a Docker file, and Docker does the rest.

Here's what you need to know:

- **Run containers**: Docker lets you run these little packages called containers. It's like running tiny virtual computers just for your app.
- **Inspect active containers**: You can peek inside running containers to see what's going on. Handy for debugging.
- **Docker Networking**: Docker helps containers talk to each other. Think of it like setting up a phone line between them.
- **Persist data with Docker Volumes**: Docker can keep your data safe even if a container shuts down. It's like having a secret vault for your stuff.
- **Dockerize apps using Dockerfiles**: Write a simple file, and Docker magically turns it into a container. It's like a recipe for your app.
- **Run multiple containers using Docker-Compose**: Docker makes it easy to run lots of containers at once. It's like managing a whole fleet of little ships.
- **Work with Docker Repository**: Docker can store your containers in a special place called a repository. It's like having a library for your apps.

Step 7: CI/CD Tools

In today's fast-paced development world, manual tasks are a waste of time, money, and resources. As a DevOps Engineer, you need to automate and streamline workflows using CI/CD tools. Here's what you should learn:



What to Learn:

- **Writing Automation Scripts**: Learn to write scripts that automate testing and deployment processes.
- **Setting up Monitoring & Feedback**: Understand how to set up monitoring systems to give feedback on the performance of your applications.
- **Best Practices**: Learn the best practices for implementing CI/CD pipelines.
- **Notifications Setup**: Set up notifications to channels like Slack or Discord to keep your team informed about the status of deployments and tests.

Tools to Learn:

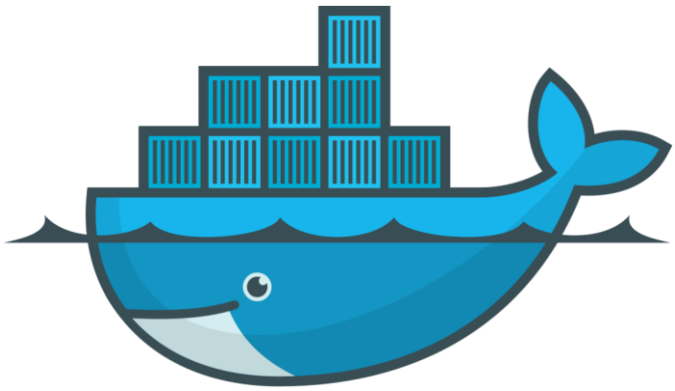
- **GitLab CI**: A tool for automating the testing and deployment of code hosted on GitLab.
- **Jenkins**: A widely used automation server for building, testing, and deploying software.
- **GitHub Actions**: GitHub's built-in CI/CD solution for automating workflows directly from your GitHub repository.
- **Circle CI**: A CI/CD platform that automates the software development process.

Step 8: Container Orchestration

In this step, we'll talk about container orchestration, which is a way to manage and scale containers easily.

What is Container Orchestration?

Imagine you have lots of containers running your applications. Container orchestration tools help you manage these containers efficiently. They can create copies of containers, scale them up or down, and handle updates smoothly.



Popular Container Orchestration Tools

Kubernetes

Kubernetes is a popular open-source tool developed by Google. It's like a conductor for your containers. It automates tasks like scaling, creating copies, and managing applications running in containers. With Kubernetes, you can handle large-scale applications easily.

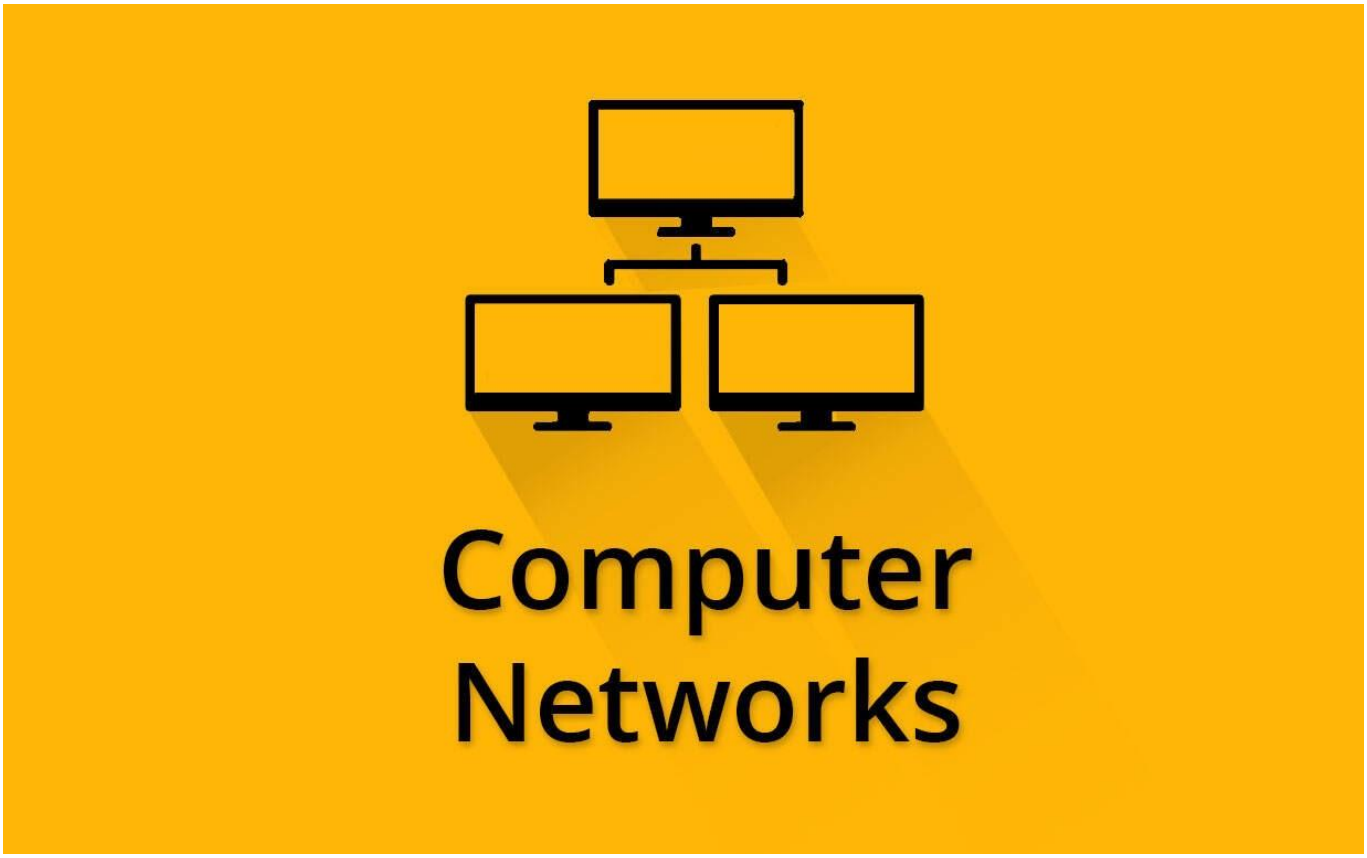
Docker Swarm

Docker Swarm is another container orchestration tool. It's simpler compared to Kubernetes and is also open-source. Docker Swarm helps you manage clusters of Docker hosts and deploy services across them.

Step 9: Networking and Security Protocols

Networking and security protocols are essential for DevOps engineers because most of their work involves servers and production environments. Learning networking concepts will help you effectively manage and troubleshoot infrastructure, deploy and manage microservices and containerized applications, automate network tasks, and handle cloud-based deployments.

Understanding these concepts bridges the gap between developers and operations teams, streamlining processes and ensuring smooth delivery.



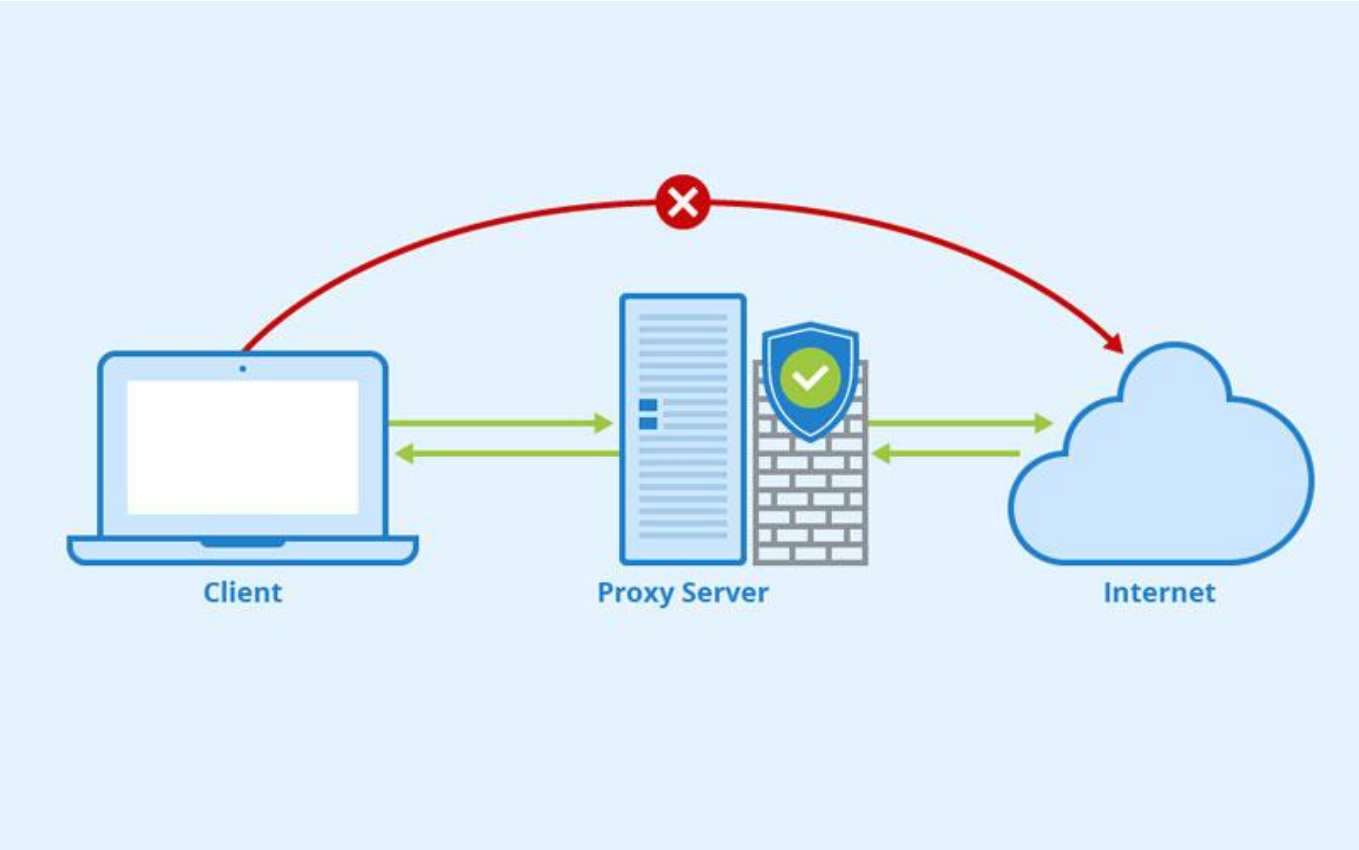
Concepts to Learn:

Here are some important concepts to focus on:

- **FTP / SFTP**: File Transfer Protocol and Secure File Transfer Protocol are used for transferring files securely over networks.
- **HTTP / HTTPS**: Hypertext Transfer Protocol and its secure version are protocols used for transmitting data over the web.
- **SSL / TLS**: Secure Sockets Layer and Transport Layer Security are cryptographic protocols used to secure communication over a computer network.
- **DNS**: Domain Name System translates domain names to IP addresses, enabling users to access websites using easy-to-remember names.
- **SSH**: Secure Shell is a protocol for securely connecting to remote servers and managing them remotely.

Step 10: Setting Up Firewalls, Proxy, and Servers

Learning how to set up firewalls, proxies, and servers is important for improving security and performance in infrastructure. Here are some key concepts to understand:



Firewall

A firewall is like a security guard for your network. It monitors and controls incoming and outgoing traffic based on predefined security rules. Learning about firewalls helps in protecting your servers from unauthorized access and cyber threats.

Proxy

A proxy server acts as an intermediary between clients and servers. It helps in managing and filtering requests, improving performance, and enhancing security. Knowing how to set up proxy servers enables efficient communication between different parts of your infrastructure.

Web Servers

Web servers are software applications that deliver web content to users over the internet. Learning how to set up and configure web servers is essential for hosting websites and web applications. Understanding concepts like virtual hosts, SSL/TLS certificates, and server configurations is crucial for ensuring smooth web server operation.

Step 11: Understand Serverless

Now, let's talk about serverless computing, a key part of the DevOps journey. Serverless computing is like magic where you don't have to worry about managing servers. Instead, you focus only on writing and running your code.

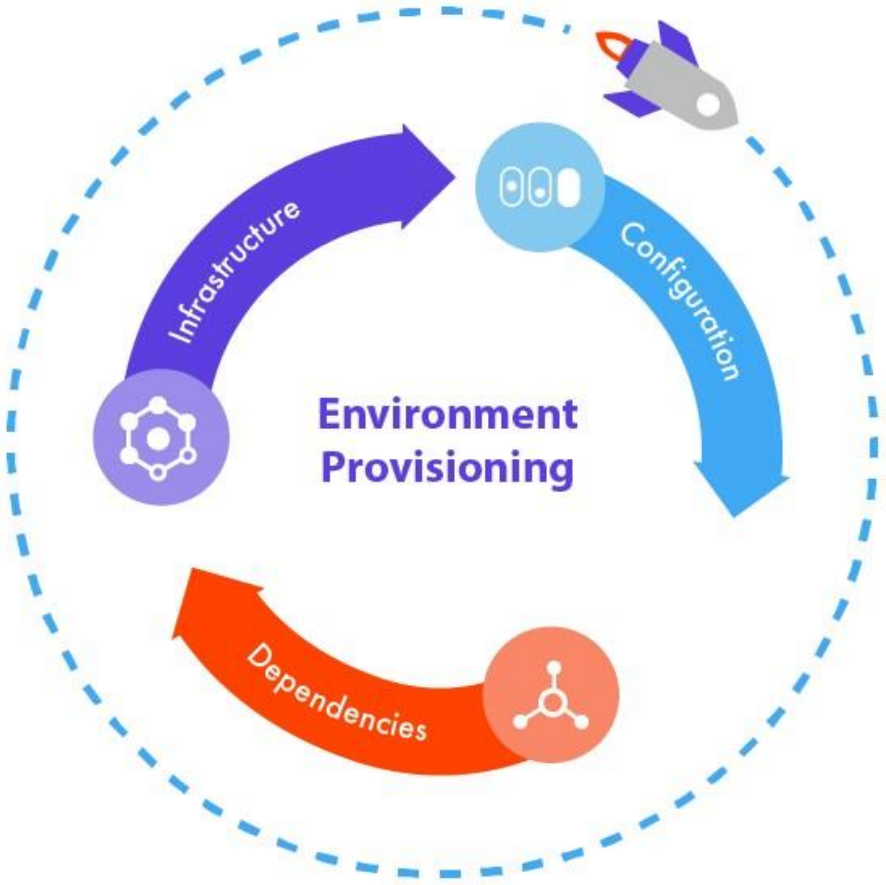
What to Learn

Here are some important things for a DevOps engineer to understand about serverless:

- **Cloudflare**: A service that helps make websites faster and more secure.
- **AWS Lambda**: A tool from Amazon Web Services (AWS) that lets you run code without managing servers.
- **Azure Functions**: Similar to AWS Lambda but part of Microsoft's cloud platform, Azure.
- **Vercel**: A platform for deploying websites and applications quickly.

Step 12: Infrastructure Provisioning

As a DevOps Engineer, you'll manage and set up infrastructure, but doing it manually (ClickOps) is slow and hard to replicate as you scale up. Now, infrastructure provisioning has become easier with automation.



Why It's Important

Your job is to automate and scale resources when needed. Learning infrastructure provisioning using scripts will make your work smoother and more efficient.

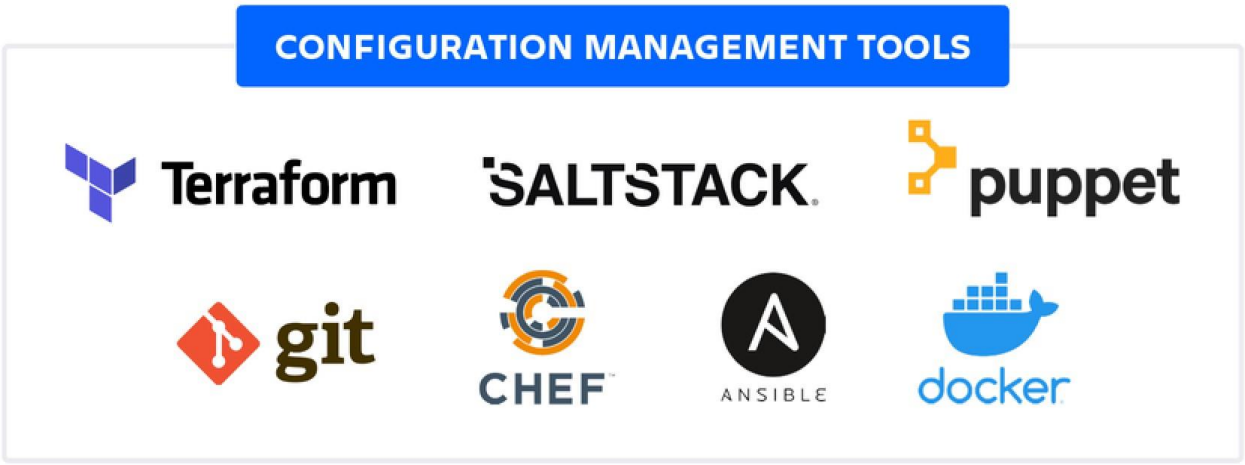
Popular Tools to Focus On

Here are some tools you should learn:

- **Terraform**: Helps you manage infrastructure as code.
- **Pulumi**: Another tool for managing infrastructure with code.
- **CloudFormation**: If you're working with AWS, this tool helps you provision and manage AWS resources.
- **AWS CDK (Cloud Development Kit)**: A newer way to define cloud infrastructure using familiar programming languages.

Step 13: Configuration Management

Configuration Management is about keeping track and controlling all the different parts of your infrastructure, software, and systems. Imagine having to set up and manage lots of servers manually, it would take forever and mistakes would happen all the time. So as a DevOps engineer, you can use tools to automate these tasks.

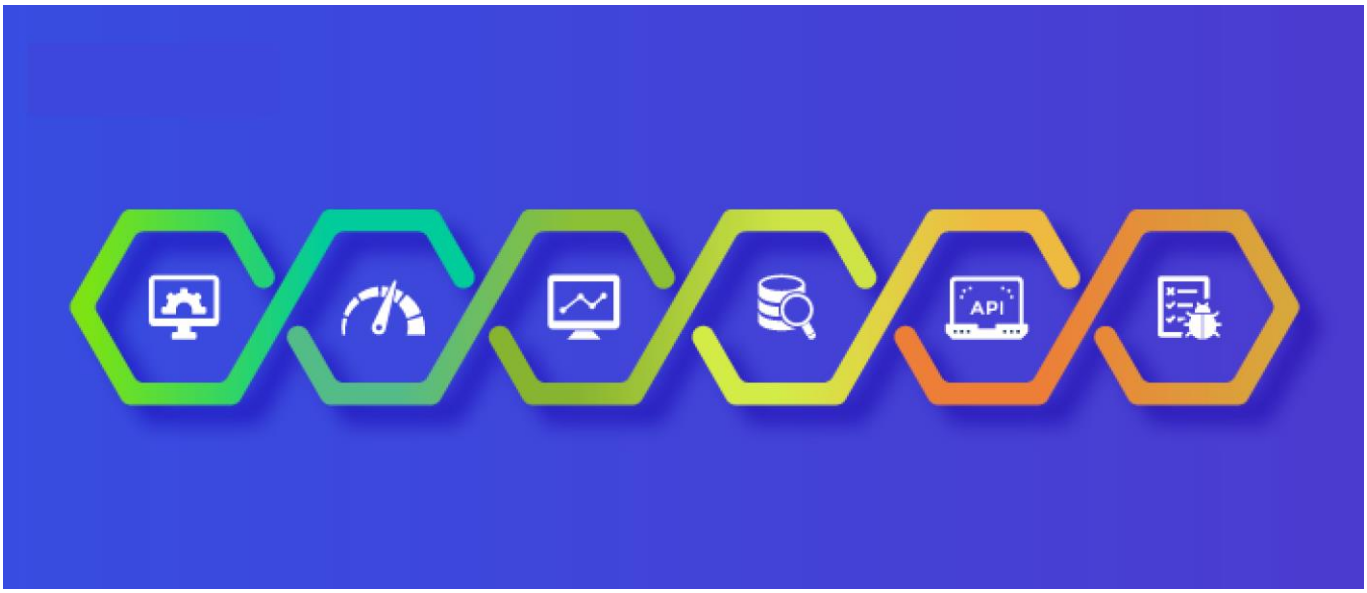


Here are some tools you should learn:

- **Ansible**: It helps automate tasks like setting up servers and managing configurations.
- **Chef**: This tool is used for automating how you set up and manage your servers.
- **Puppet**: Puppet helps in automating the configuration of your servers and software.

Step 14: Infrastructure Monitoring

Infrastructure monitoring is like keeping an eye on the health of all the parts that make your software run smoothly. It's about checking if everything is working well and fixing it if something goes wrong.



What is Infrastructure Monitoring?

It's like having a dashboard that shows you how well your systems are doing. This dashboard gathers data from different places like system logs and performance metrics. With this data, you can see if there are any problems and fix them before they become big issues.

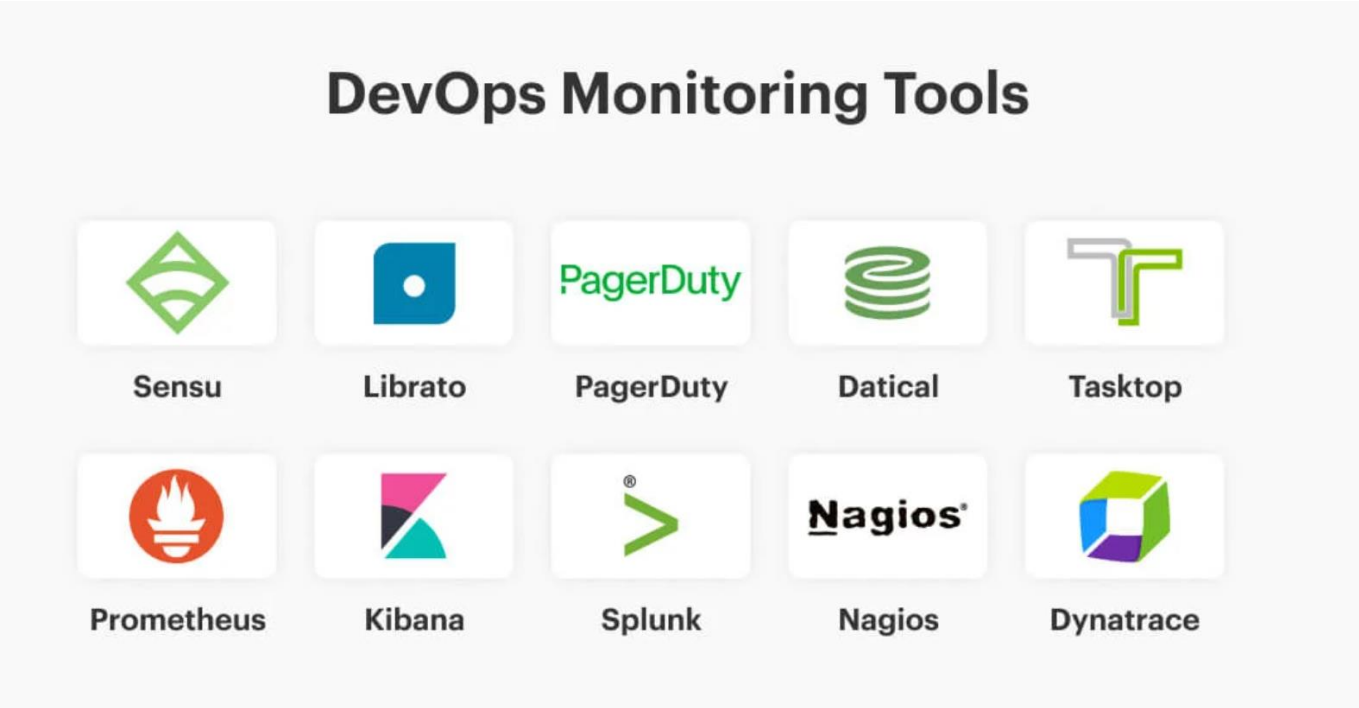
Tools You Should Learn

Here are some tools you can use to monitor your infrastructure:

- **Grafana**: Helps you visualize and understand your system's performance.
- **Datadog**: Gives you insights into your applications and infrastructure.
- **Prometheus**: Monitors your systems and generates alerts when something goes wrong.
- **Zabbix**: Helps you track the health of your servers and network devices.

Step 15: Application Monitoring

Application monitoring is important because when we make changes or fixes to our application and deploy them, there's a chance that something might go wrong. To avoid such situations, it's crucial to continuously track, measure, and analyze various metrics related to our application's performance.

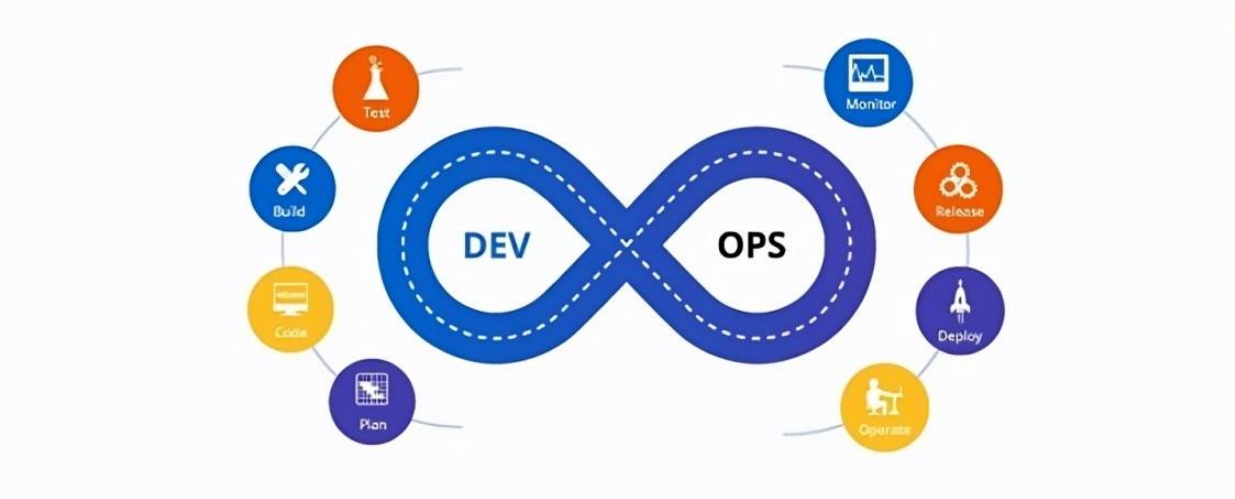


Tools for Application Monitoring

Here are some tools that help in monitoring applications:

- **Datadog**: It's a tool that gives real-time visibility into our infrastructure, performance, and logs. This helps us stay on top of any issues and performance bottlenecks.
- **New Relic**: This is a cloud-based tool that helps organizations monitor and optimize their applications. It provides insights into the performance of our applications running in the cloud.
- **Jaeger**: Jaeger is a monitoring tool specifically designed for distributed microservices architectures. It gives us deep insights into the performance of different microservices across our application.
- **OpenTelemetry**: This is an open-source framework that provides standardized instrumentation, APIs, and libraries for collecting and monitoring application performance data.
- **AppDynamics**: It's an application performance monitoring (APM) solution that allows us to monitor, analyze, and optimize the performance of our applications in real-time.

Step 16: Other DevOps Roadmap Topics to Learn



Logs Management

Logs management is about dealing with the data produced by applications, systems, and infrastructure.
You collect, store, analyze, and visualize this data to understand what's happening in your systems.

Artifact Management

Artifact management involves handling and versioning software components like binaries, libraries, and configuration files. You use tools to store and share these artifacts, ensuring consistency and traceability in your software builds and deployments.

Artifactory

Artifactory is a tool by JFrog that helps manage and distribute software artifacts. It supports various package formats and technologies.

Nexus

Nexus Repository Manager, provided by Sonatype, is another tool for managing software artifacts. It supports formats like Maven, npm, and Docker.

Service Mesh

In a setup with many small services or components, a service mesh helps them communicate. It's like a layer in your infrastructure that makes sure everything can talk to each other.

Cloud Design Patterns

Cloud design patterns are reusable solutions for common problems in cloud-based applications. They help you design and build applications that are scalable, available, and resilient in cloud environments.

Exploring Career Opportunities with DevOps Skills

Learning various tools gives you an advantage because of your diverse knowledge. It opens up opportunities for different roles, including the core DevOps Engineer role.

DevOps Engineer

As a DevOps engineer, your main job is to automate and improve the software development process. You work with different teams to implement practices like continuous integration and delivery. In India, DevOps Engineer salaries range from ₹6 to ₹10 lakhs per year.

Site Reliability Engineer (SRE)

SREs focus on making sure large systems run smoothly. They use DevOps principles to design resilient systems and automate incident response. In India, SRE salaries range from ₹7 to ₹19 lakhs per year.

Cloud Engineer

Cloud engineers manage cloud infrastructure using DevOps skills. They work with platforms like AWS, Azure, and Google Cloud to build scalable applications. In India, Cloud Engineer salaries range from ₹5 to ₹8 lakhs per year.

System Administration

DevOps Engineers are skilled in system administration, managing servers, networks, and storage systems.
In India, System Administration salaries range from ₹4.5 to ₹6.5 lakhs per year.

Software Developer

With DevOps knowledge, you can automate development processes. This gives you an edge as a software developer. In India, Software Developer salaries range from ₹5 to ₹8 lakhs per year.

Test Automation Engineer

Test Automation Engineers write scripts to automate testing procedures. In India, their salaries range from ₹4 to ₹6 lakhs per year.

Network Engineer

Network Engineers design and manage network architectures for cloud-native applications. In India, Network Engineer salaries range from ₹4 to ₹5.5 lakhs per year.

Advice for Beginners with Limited or No IT Background

If you're new to IT or have limited experience, follow this roadmap to become a DevOps Engineer:

1. **Start with the Basics**:
    - Focus on mastering the fundamentals of IT.
    - Learn about basic computer concepts and how systems work.
2. **Learn Linux**:
    - Linux is essential for DevOps.
    - Get comfortable with using Linux commands and understanding its file system.
3. **Use Documentation**:
    - Don't underestimate the power of documentation.
    - Practice reading and understanding technical documentation.
    - It's a skill that will help you immensely in your journey.
4. **Avoid "Tutorial Hell"**:
    - While tutorials are helpful, don't get stuck only following them.
    - Practice what you learn and try to apply it in real-world scenarios.
    - Don't be afraid to make mistakes; they're part of the learning process.
5. **Build Projects**:
    - Hands-on projects are key to understanding concepts deeply.
    - Create projects that utilize the technologies and tools you're learning about.
    - Building projects will solidify your understanding and boost your confidence.
6. **Seek Guidance**:
    - Don't hesitate to ask for help from experienced professionals.
    - Connect with seniors in the field and seek mentorship.
    - Join communities and forums where you can interact with others on a similar journey.

Remember, becoming a DevOps Engineer is a journey that requires patience, dedication, and continuous learning. Take it one step at a time, and don't be discouraged by challenges along the way. Keep pushing forward, and you'll reach your goals!

Summary of DevOps Roadmap

Getting the Prerequisites Right

Before diving into DevOps, make sure you understand these basics well:

- **Networking Concepts**: Understand how computers communicate with each other.
- **Operating Systems**: Learn about how computers work internally.
- **Data Structures and Algorithms**: Know how to organize and manipulate data efficiently.

Automation

Automating processes like delivery, testing, and deployment is essential in DevOps. This makes sure things happen quickly and consistently.

Important Concepts to Focus On

Here are some key areas to focus your learning on:

- **Linux**: Operating system used widely in the industry.
- **Cloud**: Understand cloud platforms like AWS, Azure, or Google Cloud.
- **Docker**: Learn about containerization for easier software deployment.
- **Kubernetes (K8s)**: Understand how to manage containerized applications at scale.
- **CI/CD (Continuous Integration/Continuous Deployment)**: Automate the process of delivering software.
- **Monitoring**: Learn how to monitor applications and infrastructure for issues.

Keep Learning

DevOps is always evolving, with new tools emerging all the time. Keep learning about new tools and how to use them effectively. Documentation is your best friend for this!