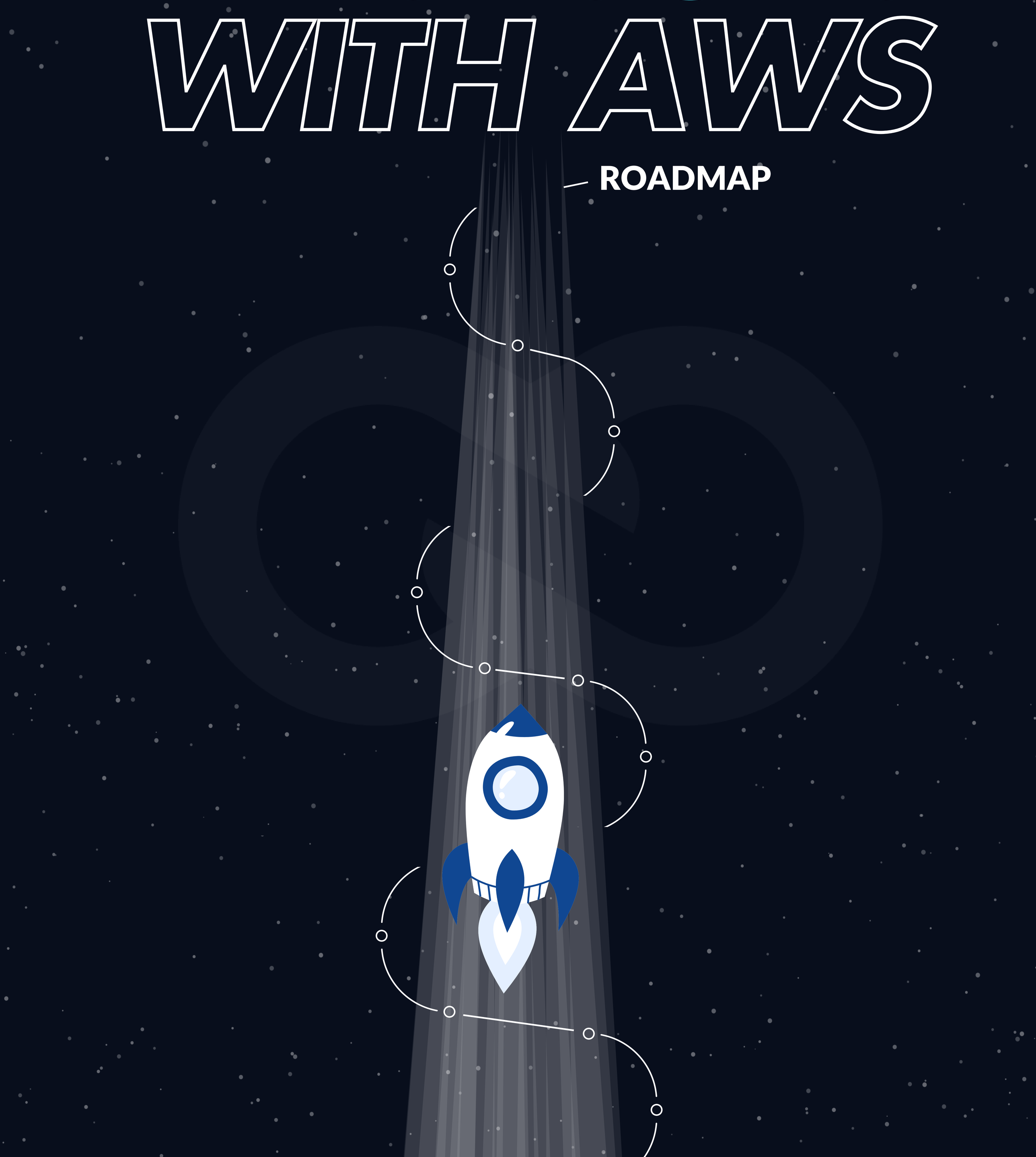
joindevops.com

DEVSECOPS WITH AWS

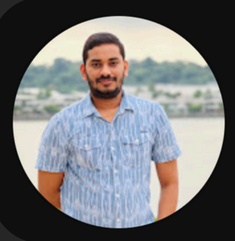
ROADMAP



joindevops.com

JPMorgan Chase & Co. DBS accenture

GOVTECH SINGAPORE



DevOps and Cloud with Siva

@DevOpsAndCloudWithSiva · 57.2K subscribers · 301 videos

My name is Sivakumar Reddy. I have total 10+ years of experience as DevOps and Cloud ...more

joindevops.com

Subscribed

12+ Years Experience

6500+ Students Trained

Sivakumar Reddy Mettukuru

Senior Cloud Architect & Expert

Sivakumar Reddy M., Founder & CEO of Joindevops & Linuscode Technologies, is an expert in strategic leadership with extensive experience in DevOps, cloud computing, Java development, IoT, telecom, banking, media, and R&D. He specializes in modern cloud-native application development, guiding businesses through legacy-to-cloud migrations, and ensuring robust cloud security and cost optimization. Sivakumar is a leader in integrating DevSecOps practices across projects and excels at optimizing performance in modern architectures. With a passion for exploring advancements in cloud technologies and modern development practices, he is committed to driving innovation and delivering cutting-edge solutions for his clients.

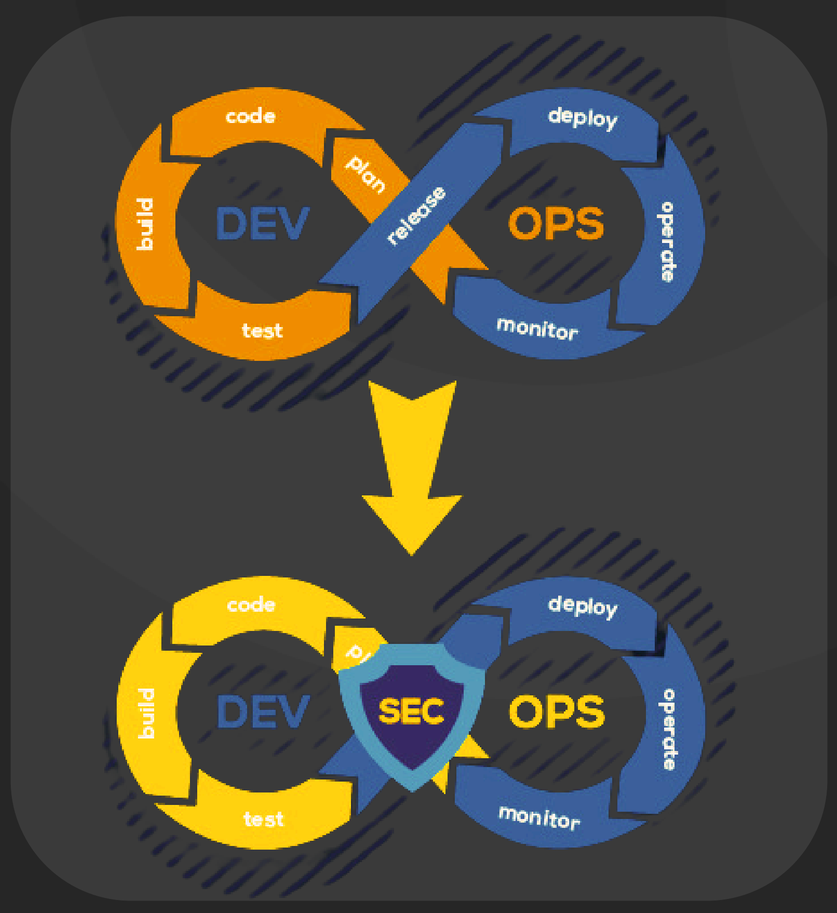
ఈనాడు Outlook సాక్షి mint Deccan Chronicle

www.joindevops.com/devsecops-with-aws-siva-86s

DevOps vs DevSecOps

DevOps and DevSecOps are both practices that aim to enhance the software development and deployment process, but they have distinct focuses and approaches. Let's break down the key differences between them:

Aspect	DevOps	DevSecOps
Focus	Speed and efficiency	Security integrated into DevOps
Goal	Continuous delivery	Secure continuous delivery
Security	Security handled towards the end	Security embedded throughout the process
Team Collaboration	Dev + Ops	Dev + Sec + Ops
Tools Used	Jenkins, Docker, Kubernetes	Jenkins, SonarQube, Veracode, Kubernetes with security focus



Why Choose DevSecOps?

As organizations become more reliant on technology and data, the security of software has become increasingly important. DevSecOps makes security a collaborative, continuous, and automated part of the DevOps process, leading to better, safer software.

In a world of increasing cybersecurity threats, DevSecOps represents the next step in the evolution of DevOps, helping teams not only deliver software fast but also deliver secure software without compromising on speed or quality.

Tools and Technologies

Covered in this course

aws

kubernetes

Linux

BASH
THE BOURNE-AGAIN SHELL

Jenkins

ELK

git

docker

HashiCorp
Terraform

ANSIBLE

Prometheus

Grafana

python

sonarqube

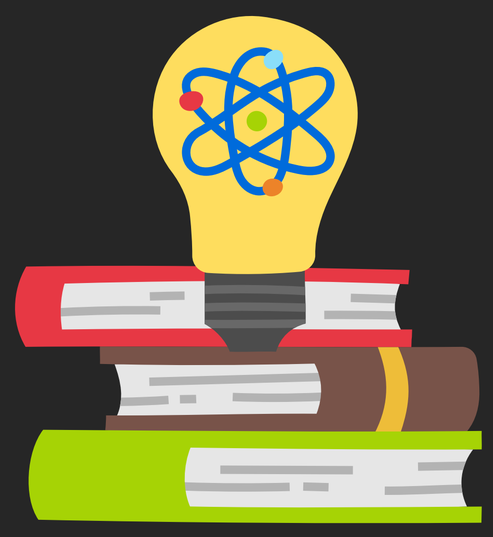
argo

sonatype
nexus repository

HELM

VERACODE

Ready to dive in? Join us at JoinDevOps.com and take the first step toward mastering DevSecOps with AWS!



www.joindevops.com/devsecops-with-aws-siva-86s

Start Here

This is a **step-by-step roadmap crafted from the perspective of a DevSecOps professional and instructor**. It outlines the most effective path for becoming a DevSecOps expert with AWS, built on practical insights and real-world experience.

With this roadmap, I hope to guide you on this rewarding yet challenging journey into DevSecOps.

We've designed this course to make it more personalized based on your current background as you transition into DevOps or DevSecOps. So whether you're coming from:

- System Administrator
- Software Developer
- Test Automation Engineer
- Network Engineer
- Individuals with limited or no IT knowledge

After completing the DevSecOps roadmap, you'll gain insights into how to begin your DevSecOps journey based on your unique background.

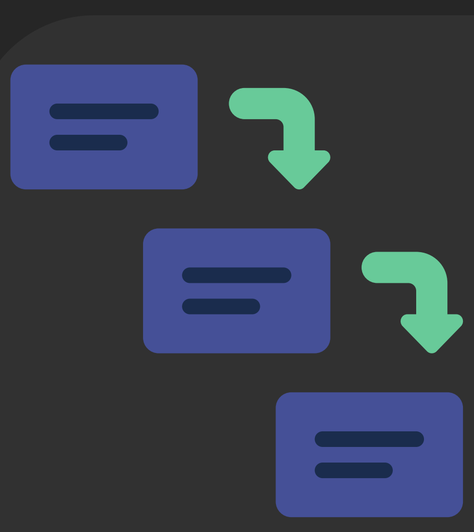
One important thing to keep in mind:

DevSecOps covers the entire software development lifecycle, which involves working with numerous tools and technologies. DevSecOps is constantly evolving, with new tools being introduced regularly.

Understanding Software Development

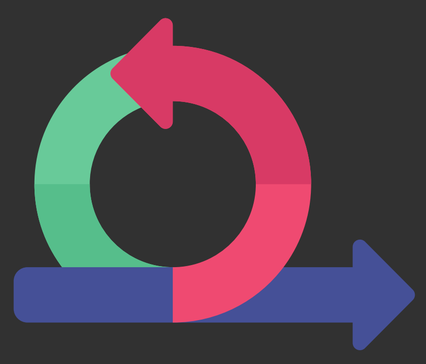
As a **DevOps professional**, you won't necessarily be writing code, but you'll work closely with development teams to improve and automate their processes. To be effective, it's essential to understand the **Software Development Life Cycle (SDLC)** and why DevOps has become vital in modern development.

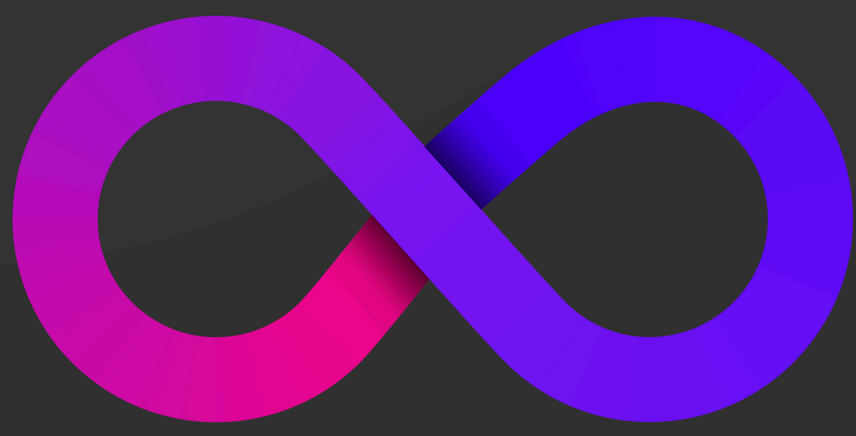
And generally understand what the whole software development lifecycle covers from idea to code, all the way to releasing it to the end users!



Waterfall Method:
A traditional, linear approach where each stage of development must be completed before the next begins.

Agile Methodology:
A more dynamic, iterative approach where development happens in sprints, allowing for frequent adjustments based on feedback.





DevOps combines the speed of Agile with the operational focus needed for reliable deployments, ensuring quality and stability in the software delivery pipeline.

Linux and OS Basics

In **DevOps**, part of your job is to set up and look after the infrastructure where applications run. To do this well, you need to be comfortable with some basics of server management, especially on Linux, since that's the main OS used in most setups.

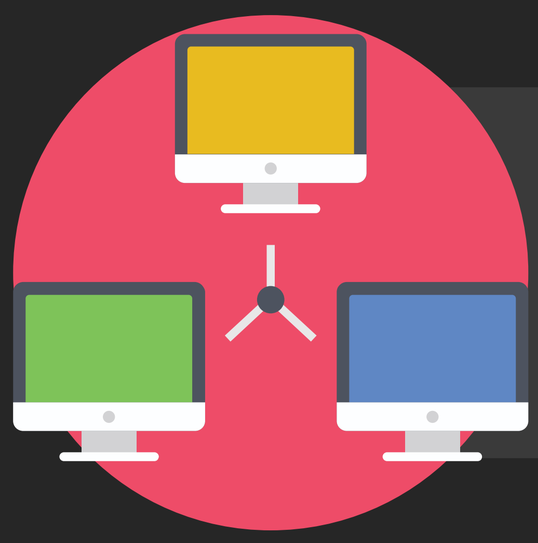
Basic concepts of Operating Systems, you need to understand:

- Linux Commands
- Linux File System & Permissions
- Editors
- Package & Network Management
- SSH Key Management
- User and Service management

Since most servers use **Linux,** being comfortable with the Command Line Interface (CLI) will make your life a lot easier. We mostly use RedHat Linux throughout the course

To set up and secure your infrastructure, it helps to have a basic understanding of networking and security. Here are a few things you'll learn:

- Basics of IP addresses, ports, and DNS
- HTTP/HTTPS
- Security Groups
- Launching EC2 Machine



What is Computer?

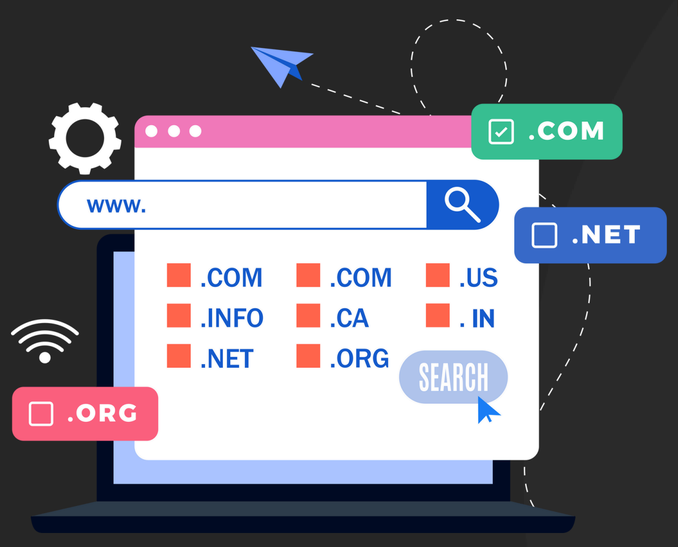
Client - Server Architecture

Linux Advantages over windows

Quick Tip: You don't need to be a full-on **SysAdmin** for **DevSecOps**. Just focus on the essentials – leave the deeper server management to the pros. Your goal is to know enough to get things up and running securely.

Manual Deployment of 3-tier Web Application

Here, we're diving into a hands-on project that will teach you how to manually deploy a fully functional **3-tier web application**. This is a powerful learning experience because you'll be setting up every layer of the application by yourself, from the database to the backend and frontend – all hosted on your own custom domain!

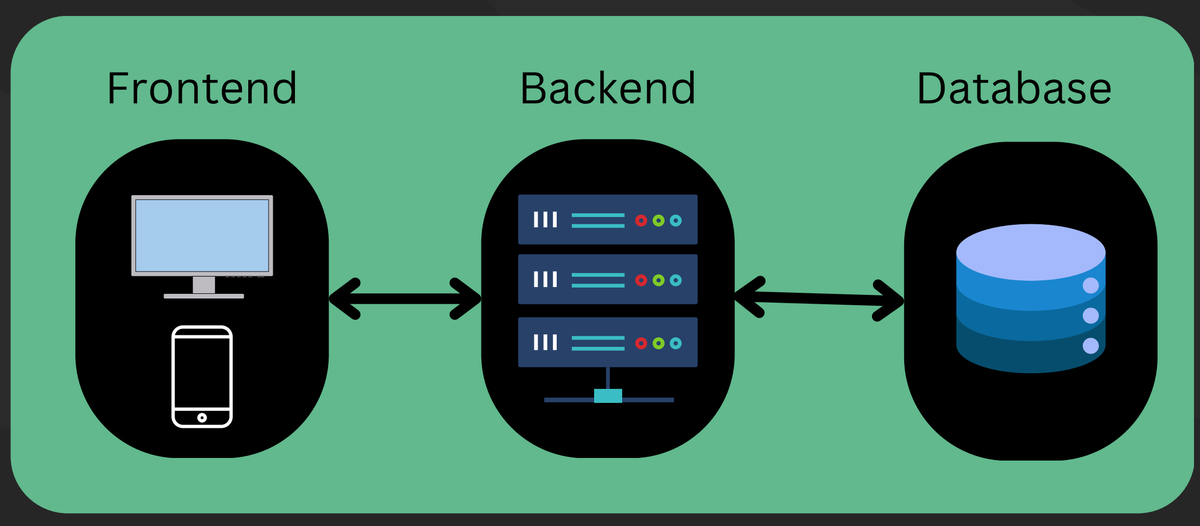


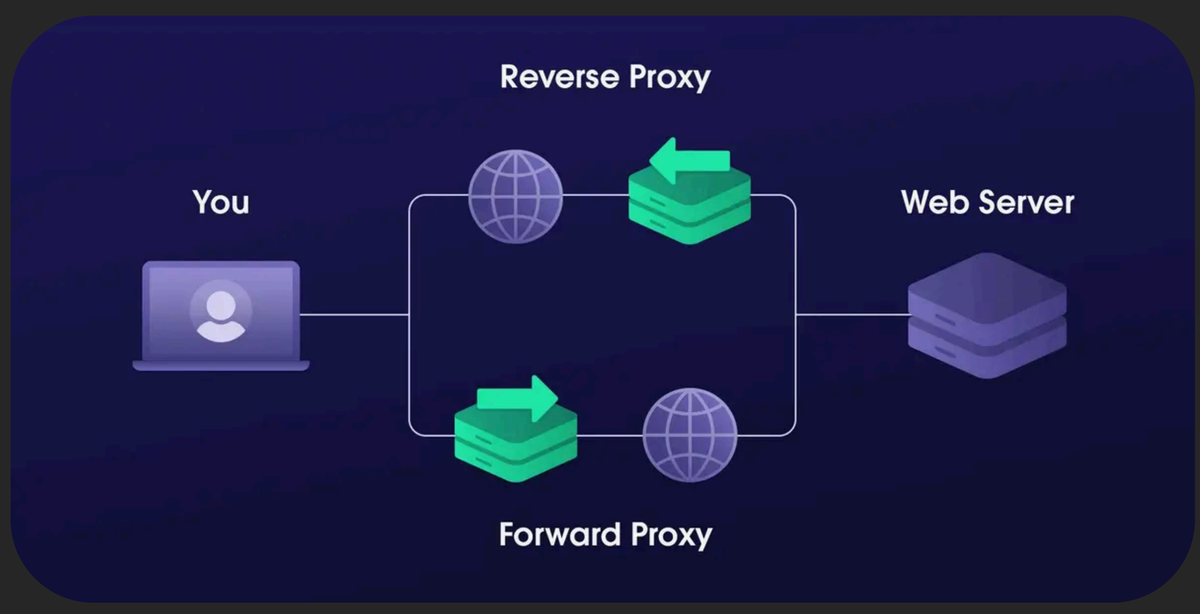
Understanding the "Why" Behind Manual Deployment:

Deploying everything manually might seem like extra work, but it's essential to understanding how your application really works. By going through each setup step yourself, **you'll gain a deeper knowledge** of what's happening under the hood. This will make it much easier to automate these processes later using **DevOps tools**.

Here's What You'll Be Doing:

- Launching Your Server
- Purchasing and Setting Up Your Own Domain (www.yourname.com)
- Manual Deployment of the 3-Tier Application
  a. Database
  b. Backend
  c. Frontend
- Exploring Networking Concepts
  - Forward & Reverse Proxy
  - Linux Folder structure
  - Soft link and Hard link





Shell Scripting

Since you are closely working with developers and system administrators to also automate tasks for development and operations, you will need to write scripts and small applications to automate them.

Shell scripting is like creating a list of instructions for your server to follow, which saves you time and reduces the chances of mistakes.

You'll start with the basics and work your way up to industry-level techniques. By the end of this section, you'll know how to automate the setup and deployment of your own 3-tier web application.

What You'll Learn:

Shell Scripting Basics and Advanced Concepts

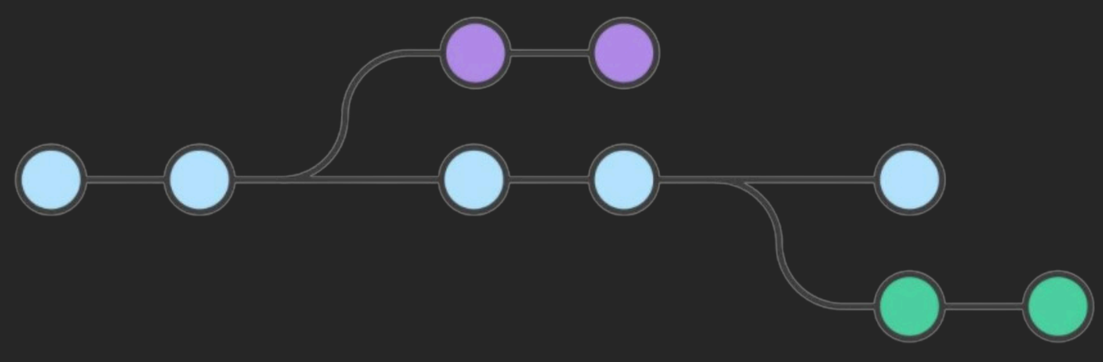
- **Special Variables**
- **Conditions**
- **Functions**
- **Colors**
- **Exit Status**
- **Loops**
- **Logs and Redirectors**
- **Idempotency**
- **Monitoring and Backup scripts**

Version Control with Git

You'll also learn about Git, which is a tool that helps you keep track of changes in your scripts. We'll cover:

- GitHub Setup: How to set up an account and create a repository to store your scripts.
- Key Commands:
  - **git add**: Add changes to be committed
  - **git commit**: Save your changes with a message describing what you did
  - **git push**: Upload your changes to GitHub

This will allow you to manage different versions of your scripts, collaborate with others, and keep a history of your work.



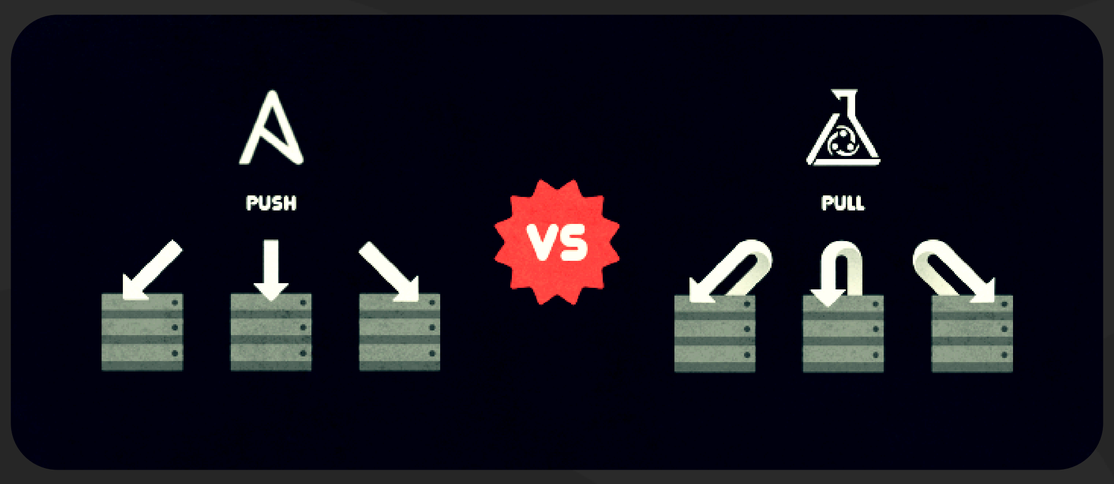
Automating Deployments with Ansible

Ansible—a powerful tool that simplifies and automates the management of servers and deployments. Ansible is great for handling complex tasks with ease, making it a must-have skill for anyone looking to streamline operations and increase efficiency in a DevSecOps role.

Why Move Beyond Shell Scripts?

Shell scripting is powerful, but it has its limits. When managing lots of servers or handling complex configurations, shell scripts can become cumbersome, difficult to maintain, and prone to errors. That's where Ansible comes in.

Push vs. Pull Architecture



Hands-On:

Writing Playbooks to Automate Deployments

- You'll start by writing simple playbooks that automate common tasks.
- Then, we'll move on to writing playbooks that automate the deployment of your 3-tier web application. You'll automate the setup of the database, backend, and frontend, all with a few lines of YAML.
- Roles, Tags, Vault, Dynamic Invenory, Handlers, etc.

By the end of this module, you'll know how to use Ansible to automate the setup and deployment of your entire web application infrastructure. You'll also have a deeper understanding of how automated tools make life easier, from simplifying repetitive tasks to ensuring consistency across environments.

Infrastructure as Code (IaC) with Terraform

Manually creating and maintaining infrastructure is not only time-consuming but also prone to errors—especially when you need to replicate environments, like Development, Testing, and Production. To overcome these challenges, we use Infrastructure as Code (IaC), where we write code to manage and configure infrastructure. This approach brings consistency, automation, and reliability.

Why Terraform Over Ansible for Infrastructure Provisioning?

Ansible is a powerful tool for managing configurations, but it has limitations when it comes to provisioning complex infrastructure. Terraform, on the other hand, is purpose-built for provisioning infrastructure, which makes it more efficient for tasks like setting up cloud resources, networking, and scaling. Here's how they differ:

- Ansible: Great for configuration management—think of setting up software and managing server settings.
- Terraform: Ideal for provisioning—creating and managing cloud infrastructure such as servers, databases, and networking.

Learning Terraform from Basics to Advanced Concepts
We'll start by understanding the basics of how Terraform works, and we'll move step-by-step into more advanced topics. Here's what you'll learn:

- File and Folder Structure
- Learn how Terraform's files and folders are organized.
- Writing Custom Modules
- Modules are reusable pieces of Terraform code.

HashiCorp Terraform

1. Key Terraform Commands
2. To manage infrastructure effectively, you'll use commands such as:
   - terraform init
   - terraform plan
   - terraform apply
   - terraform destroy
3. Advanced Commands and Concepts
   - Upgrade & Format (upgrade, fmt)
   - State Management (show state, import, taint, remote state, state locking)
   - Variables (tfvars, locals) configurations reusable and more readable.
   - Workspaces
   - Loops and Conditionals
   - Provisioners
4. Hands-On: Creating Infrastructure and Deploying the Web Application
5. Once you understand the concepts, you'll create infrastructure for your 3-tier web application:
   - Set up networking components like Virtual Private Clouds (VPCs), subnets, and security groups.
   - Configure elastic IPs, NAT gateways, and load balancers to distribute traffic efficiently.
   - Deploy your backend, frontend, and database layers with complete automation using Terraform.

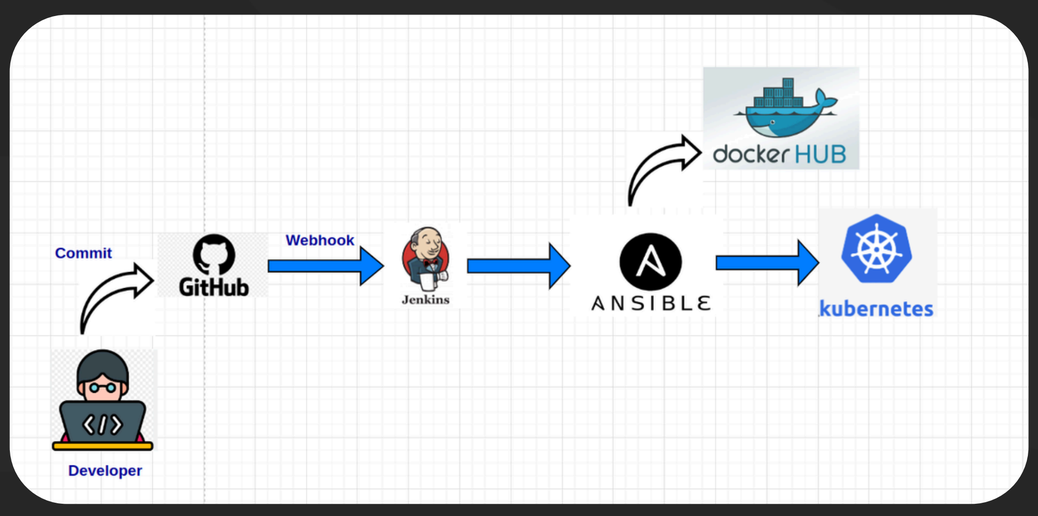
CI/CD Pipelines: Automating Deployments with Jenkins/ GitHub Actions

CI/CD (Continuous Integration and Continuous Deployment) is at the core of DevOps. It helps automatically build, test, and deploy new code changes, making the process fast, repeatable, and reliable. When developers make changes—like adding a feature or fixing a bug—the CI/CD pipeline ensures that the updated code gets tested and deployed seamlessly.

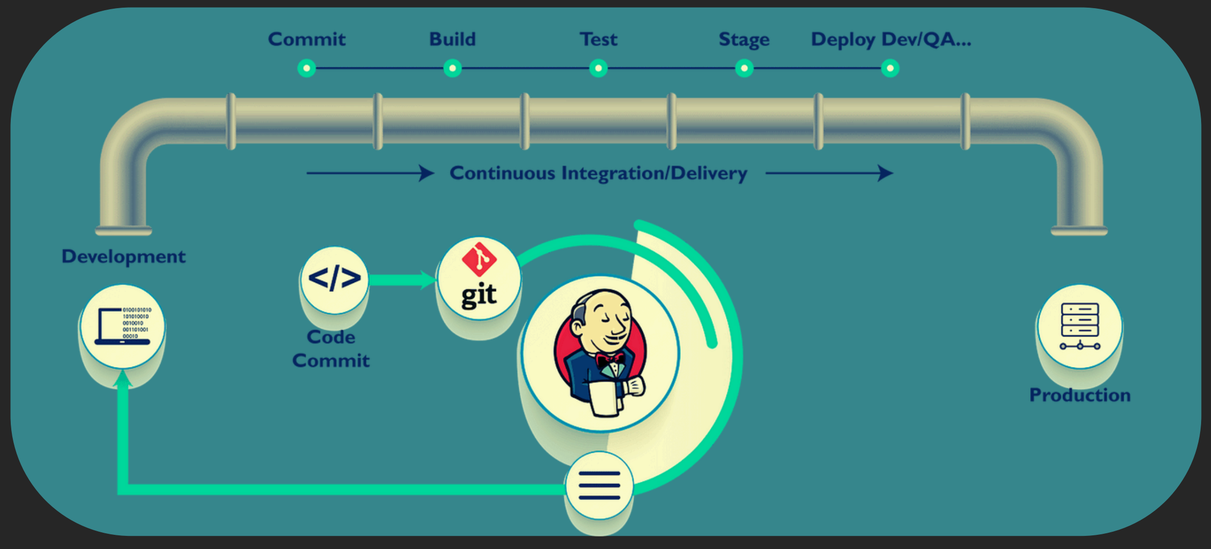
Key Topics We'll Cover:

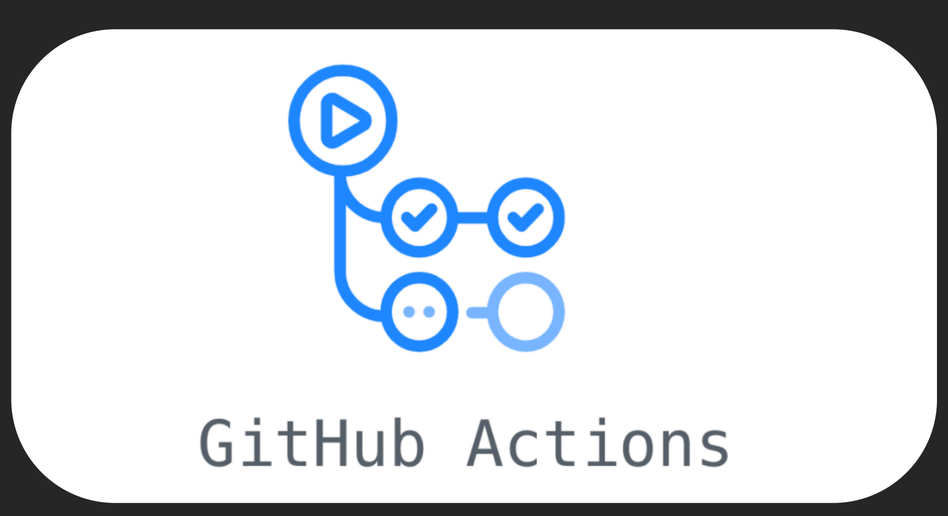
- Jenkins Pipeline Jobs: Learn to create jobs to automate testing and deployment of code.
- Master-Agent Setup: Understand how Jenkins' Master-Agent setup allows scalability.
- SonarQube Integration: Use SonarQube to automatically check for code quality issues and vulnerabilities.
- Static and Dynamic Security Testing: Learn how SAST (Static Application Security Testing) and DAST (Dynamic Application Security Testing) help secure your applications.

Jenkins is one of the most popular tools for setting up these pipelines. It allows you to automate each step, from testing to deploying, ensuring consistency and reducing manual errors. Alongside Jenkins, we also use tools like **SonarQube** for checking code quality and **Veracode** for security scans.



By the end of this module, you'll be able to set up automated pipelines that take code from a developer's push to deployment—all with Jenkins, ensuring high quality and secure software releases.





Cloud Provider

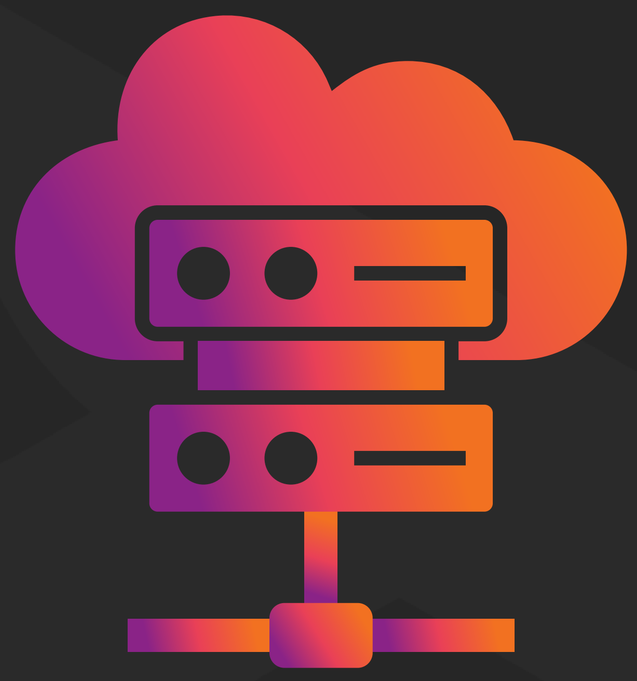
Nowadays many companies use virtual infrastructure on the cloud, instead of managing their own infrastructure. These are Infrastructure as a Service (IaaS) platforms, which offer a range of additional services, like backup, security, load balancing etc

AWS Services You'll Learn:

- VPC & Subnetting: Set up secure, isolated networks.
- Routes & Peering: Manage traffic and connect multiple VPCs.
- EC2: Deploy virtual servers for hosting applications.
- Autoscaling & Load Balancing: Scale automatically and balance traffic efficiently.
- Route 53: DNS service for directing user requests.
- IAM: Control access and manage permissions securely.
- CloudFront (CDN): Deliver content quickly worldwide.
- S3: Store and retrieve data effortlessly.
- Lambda: Run code without managing servers.
- EKS & ECR: Manage Kubernetes clusters and container images.
- KMS: Encrypt and secure your data.

Among cloud providers, AWS is the most widely used and powerful, but it can also be challenging to learn.

Other popular ones:
Microsoft Azure, Google Cloud



These services are platform-specific. So you need to learn the services of that specific platform and learn how to manage the whole deployment infrastructure on it

AWS has loads of services, but you only need to learn the services you/your company actually needs. E.g. when the K8s cluster runs on AWS you need to learn the EKS service as well.

Once you learn one IaaS platform, it's easy to learn others

Containerization with Docker

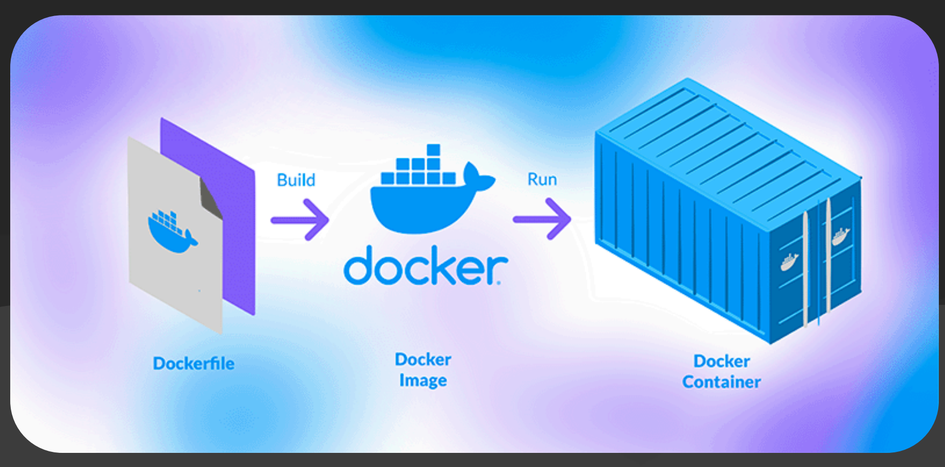
Docker has become the standard for packaging and running applications as containers. If you're building applications today, chances are you'll use Docker to deploy them efficiently. In this section, we'll explore what containers are, how Docker works, and why it's a game-changer for modern software development.

From Monolithic to Microservices:

- In the past, applications were developed as large, monolithic systems. Today, we use microservices—small, independent components that make development faster and more flexible.

Bare Metal vs. VMs vs. Containers:

- Bare Metal: Physical servers running applications directly, which makes scaling difficult.
- Virtual Machines (VMs): Virtualize the entire operating system, which makes them more flexible than bare metal but still resource-heavy.
- Containers: Containers virtualize only the application layer. This makes them lightweight and faster, as they don't need to run a full OS for every instance.



Getting Started with Docker

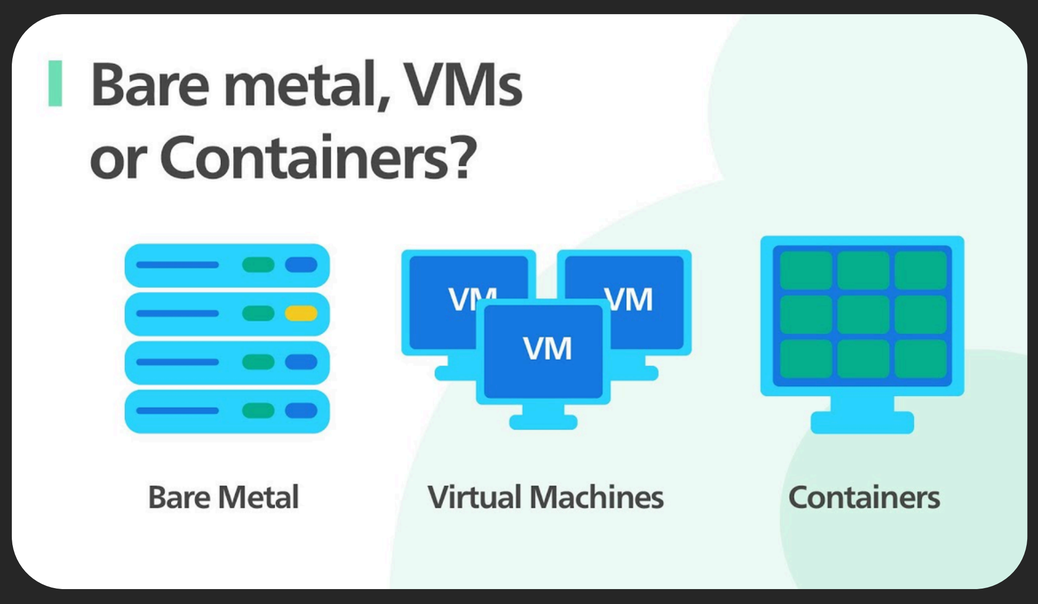
Docker is a tool that helps you create, run, and manage containers easily. It allows you to package your application and all its dependencies into a single, isolated unit called a container.

- Docker Installation and Commands: You'll learn how to install Docker and use basic commands like docker run, docker ps, docker stop, etc., to manage containers.
- Docker Networking: Understand how Docker manages networking, allowing multiple containers to communicate with each other seamlessly.
- Docker Volumes: Persist your data by attaching volumes to your containers so that information isn't lost when the container stops.
- Docker Compose: Use Docker Compose to run multiple containers together—essential for deploying more complex applications with databases, backends, and frontends.

Building a Real Project with Docker

You will set up our 3-tier web application using Docker. This includes:

- Creating Dockerfiles: Writing Dockerfiles to define how your application should be built into a container.
- Best Practices: You'll learn some best practices to ensure your containers are efficient, such as:
  - Using minimal base images to keep your containers lightweight.
  - Running containers as non-root users for better security.
  - Leveraging multi-stage builds to keep images clean and small.



Container Orchestration with Kubernetes

Why Kubernetes Over Docker Alone?

With Docker, managing a few containers is easy. But as applications grow and we need to run multiple services, scaling becomes a challenge. Kubernetes automates the deployment, scaling, and management of containerized applications, making it easier to maintain large systems. That's why we use Kubernetes alongside Docker for better scalability and reliability.

We'll use Amazon EKS (Elastic Kubernetes Service), which helps us manage Kubernetes without worrying too much about the underlying infrastructure.

Kubernetes, also known as K8s, is the most popular tool for managing containers at scale. While Docker helps us create and run individual containers, Kubernetes comes into play when you need to orchestrate hundreds or even thousands of containers across multiple servers, ensuring that everything works together seamlessly.

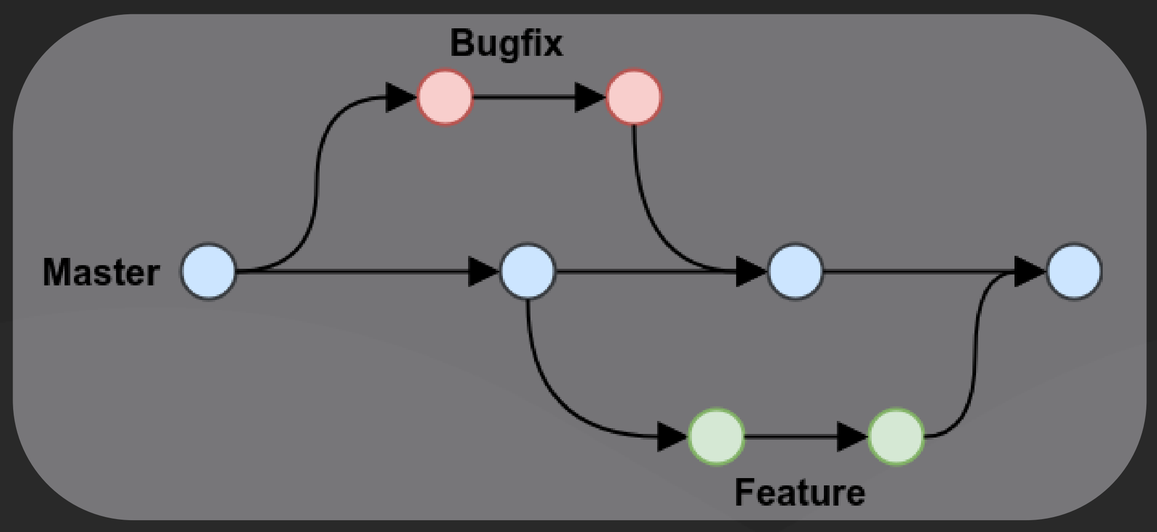
Deploying Our 3-Tier Application with Kubernetes

In this part of the course, you'll learn how to take your Docker images and deploy a complete 3-tier web application to an EKS cluster. We'll use Helm charts to simplify the deployment process.

Key Concepts You'll Learn:

1. Namespaces
   - Namespaces help organize your resources within a Kubernetes cluster, making it easy to manage and separate environments (e.g., dev, test, production).
2. Pods
   - Pods are the smallest unit in Kubernetes and represent one or more containers that run together.
   - You'll learn about Labels (for organizing resources) and Annotations (for adding metadata).
   - Set environment variables (Env) and define resource limits to manage how much CPU and memory a container can use.
3. Services
   - Kubernetes Services help expose your application to the outside world or within the cluster:
     - Cluster IP for internal communication.
     - NodePort and Load Balancer to expose your application externally.
4. Sets and Deployments
   - ReplicaSets ensure that the specified number of pod replicas are running at all times.
   - Deployments help manage updates and changes to your application.
5. Volumes for Persistent Storage
   - Understand EBS Static and Dynamic Provisioning to manage persistent storage.
   - EFS Provisioning allows sharing data between multiple pods.
6. StatefulSets
   - Use StatefulSets for applications that require stable network identities and persistent storage, ensuring that each instance of your app is uniquely identifiable.
7. Scaling Applications
   - Vertical vs. Horizontal Scaling: Learn the differences and when to use each.
   - Horizontal Pod Autoscaling (HPA): Automatically adjust the number of pods based on resource usage.
8. Helm Charts
   - Helm makes it easy to deploy complex applications using pre-configured templates. You'll learn how to use Helm charts to deploy our 3-tier application efficiently.

Version Control with Git



Why Git Matters

With Git, you can:

- **Track Changes:** Know exactly what changes were made, by whom, and why.
- **Collaborate:** Work with team members on the same project, merge code, and resolve conflicts easily.
- **Store Remotely:** Save your code centrally in a remote Git repository like GitHub or GitLab.

Key Topics You'll Learn:

1. **Merge and Rebase**
   - Merging is the process of combining changes from one branch into another. This is the default way to bring changes together.
   - Rebasing is an alternative approach that helps create a linear history, making it easier to understand. You'll learn when to use each method and the pros and cons of both.
2. **Pull Request (PR) Process**
   - A Pull Request (PR) is a way to propose changes and discuss them with your team before merging them into the main branch. PRs help ensure that code is reviewed, tested, and approved by other team members, leading to higher quality code.
3. **Resolving Conflicts**
   - Sometimes, two developers make changes to the same piece of code, which can lead to merge conflicts. You'll learn how to identify and resolve these conflicts, ensuring your code integrates smoothly.
4. **Branching Strategy**
   - Branching strategies help teams work efficiently without stepping on each other's toes. We'll cover different approaches like:
     - **Feature Branching:** Each new feature gets its own branch until it's complete.
     - **Git Flow**: A more structured strategy with branches for features, releases, and hotfixes.
     - You'll understand which strategy works best in different scenarios, keeping the project organized and manageable.
5. **Merge Strategy**
   - You'll also learn about different merge strategies:
     - **Fast-Forward Merge:** When there are no changes on the target branch, and your branch can be directly added.
     - **Three-Way Merge**: Involves combining changes when there are differences in both branches, often requiring more work but ensuring no changes are lost.

Scripting Language

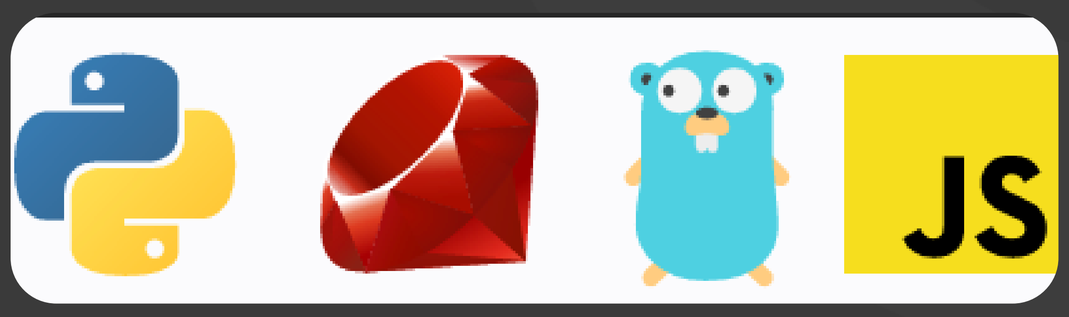
Since you are closely working with developers and system administrators to also automate tasks for development and operations, you will need to write scripts and small applications to automate them.

For that, you will need some scripting or basic programming skills.

Python is one of the most popular programming languages and easy to learn

Scripting Options:

- OS-Specific Scripting Languages:
  - **Bash, PowerShell:** Great for automating tasks within their respective environments.
- OS-Independent Languages:
  - **Python, Ruby, JavaScript, Go:** These are versatile, powerful, and more in demand in DevOps.



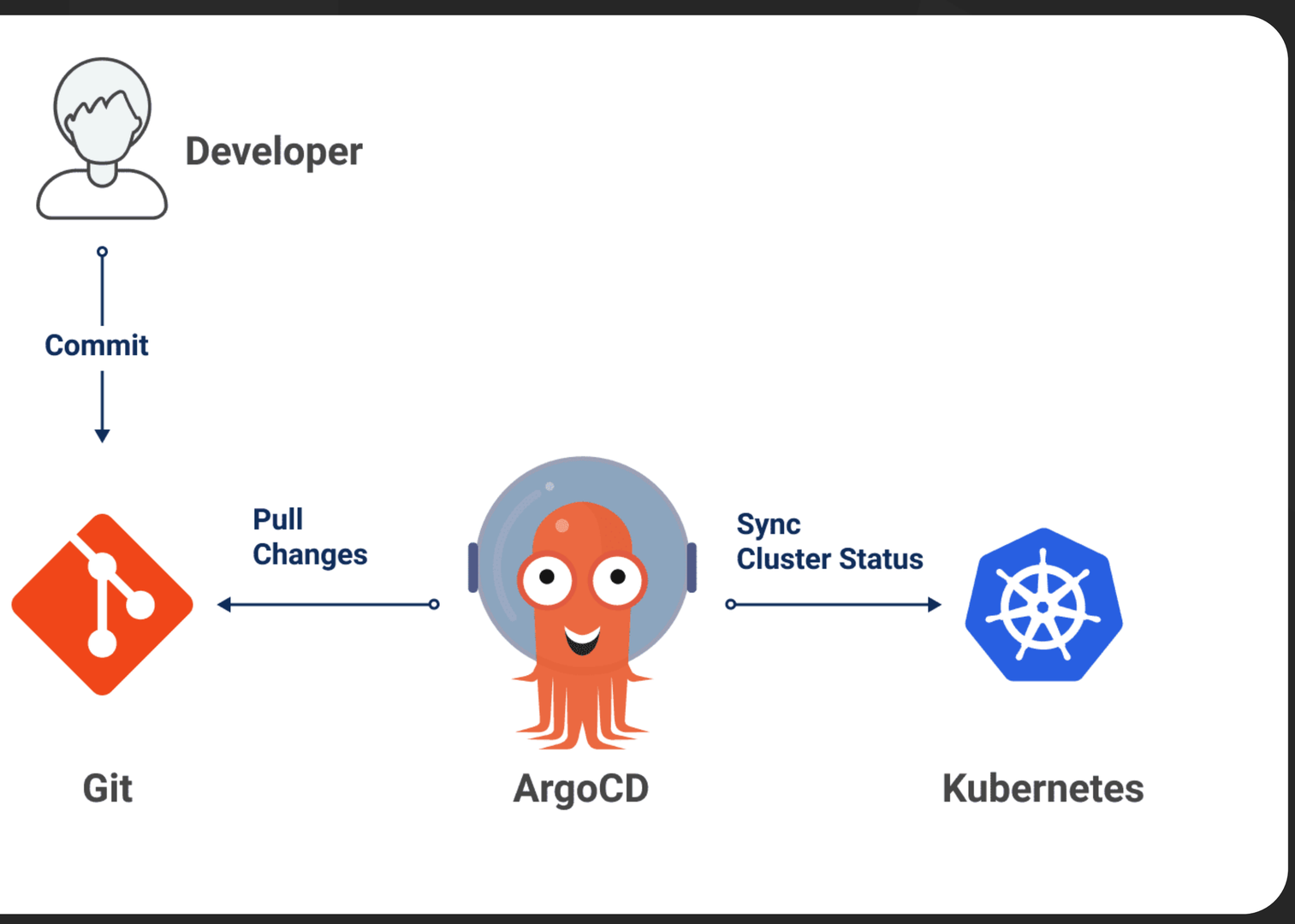
These languages are more powerful and flexible. If you know one of these, it will make you much more valuable as a DevOps engineer.

- You don't need the same level as a software developer.
- Learning how to write scripts with Python will be enough.
- And the good thing is, programming concepts stay the same, so when you learn one language well, you can easily learn new ones quite quickly.

ArgoCD

You'll start by understanding the ArgoCD architecture and how it seamlessly syncs your code to Kubernetes environments. The course will guide you through installing ArgoCD in an EKS cluster, followed by setting up and configuring it to manage multiple projects.

ArgoCD is a popular GitOps tool used for managing and automating Kubernetes deployments. It integrates directly with Git, allowing you to maintain desired application states declaratively through Git repositories.



You'll also learn to write YAML configurations for projects, set up sync policies to keep applications up-to-date automatically, and understand the best practices for using ArgoCD effectively to maintain consistency and automation in your deployments.

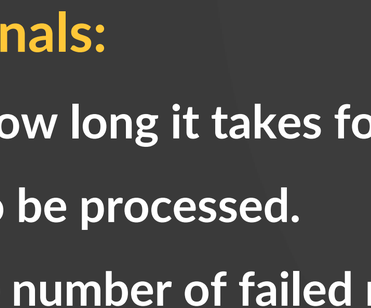
By the end of this module, you'll be able to use ArgoCD to manage Kubernetes clusters effortlessly with a clear GitOps workflow.

Monitoring & Observability

Once software is in production, it is important to monitor it to **track the performance, discover problems** in your infrastructure and the application.

4 Golden Signals:

- **Latency:** How long it takes for requests to be processed.
- **Errors:** The number of failed requests.
- **Traffic:** The overall load on the system.
- **Saturation** (e.g., CPU usage, RAM, etc.): How "full" your resources are.



Prometheus and Grafana:

- **Prometheus is a time-series database that helps collect metrics from your systems. We'll use Node Exporter to gather metrics such as CPU utilization, RAM, disk usage, and network activity.**
- **We'll also cover Prometheus Rules for creating alerts and use the Alert Manager to notify you of any critical issues.**
- **Grafana will be used to create visual dashboards for real-time monitoring of system health.**

ELK Stack for Log Management:

- **Elasticsearch helps store and search log data.**
- **Kibana provides visual insights into logs.**
- **Filebeat collects logs, while Logstash processes them to provide structured data for analysis.**

By mastering these tools and signals, you'll be able to keep a close eye on your infrastructure and applications, ensuring everything runs smoothly and proactively responding to issues before they impact users.

Good luck

on your

DevOps journey!

JOIN.DEVOPS

joindevops.com