TRAIN WITH SHUBHAM

HAPPY LEARNING

# DEVOPS ZERO TO HERO

## Course Brochure

# JUNOON

DEVOPS BY TRAIN WITH SHUBHAM


## Live DevOps training In HINDI

Job-Ready Skills to Launch Your Career in DevOps

100% Job Assistance + Real-World Projects

www.trainwithshubham.com

# Welcome to the DevOps Bootcamp in HINDI

***"The best way to predict your future is to create it."***
***– Abraham Lincoln.***

Welcome to the DevOps Bootcamp by Train with Shubham (TWS). This program is designed with a mission to not only teach you cutting-edge DevOps tools and practices but also to help you land your dream job in the tech industry.
In today's fast-paced tech world, companies are seeking professionals who can bridge the gap between development and operations. At TWS, we believe that anyone with dedication and the right guidance can master DevOps and build a promising career.

**Our Motto: Happy Learning and Get Hired Easily!**
This bootcamp is more than just a learning experience—it's a career transformation journey. You'll gain hands-on experience, solve real-world problems, and master industry tools like Docker, Kubernetes, Terraform, Jenkins, AWS, and much more.
By the end of this program, you will have not only technical knowledge but also a portfolio of projects that demonstrates your expertise to potential employers.

**Ready to Transform Your Career?**
**Join the 5,000+ students who have transformed their careers with us!**

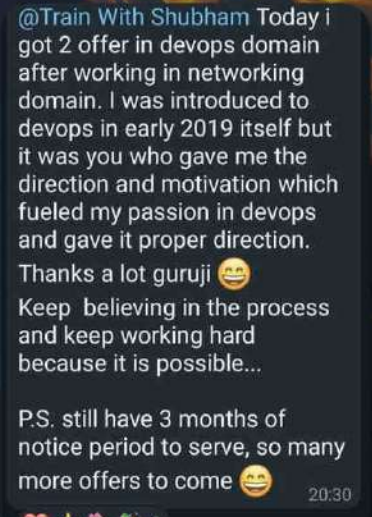
You've taken the first step by exploring this syllabus. Now it's time to take the leap and immerse yourself in a program that can truly make a difference.

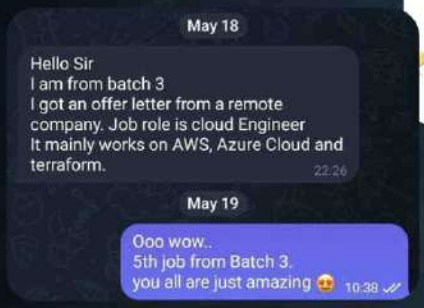
Let's start the journey!
Let's build the foundation of your DevOps career together. Remember, your success is our success.

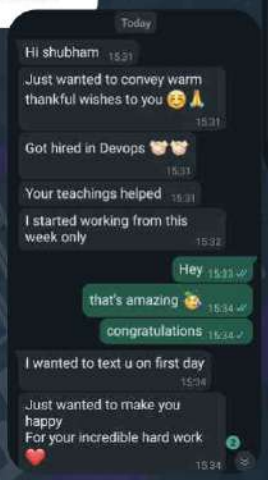
Warm regards,
**Shubham Londhe**
**Founder, Train with Shubham**

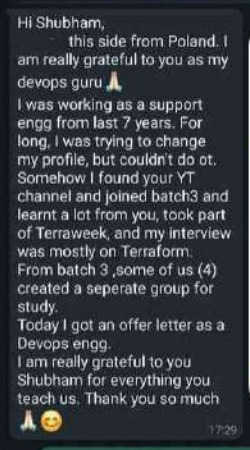
# Our Success Stories

## Previous batches have been rated 5/5 by 2500+ learners

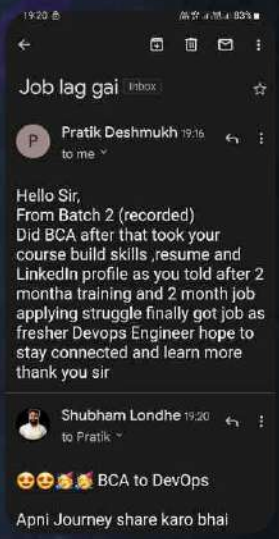



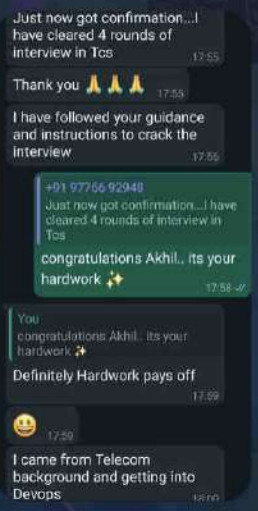



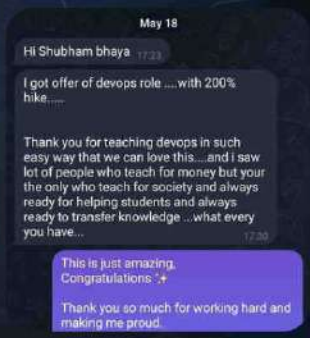










## And many more ...

www.trainwithshubham.com

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Why DevOps?

**"DevOps professionals are among the top 5 most in-demand roles globally."**

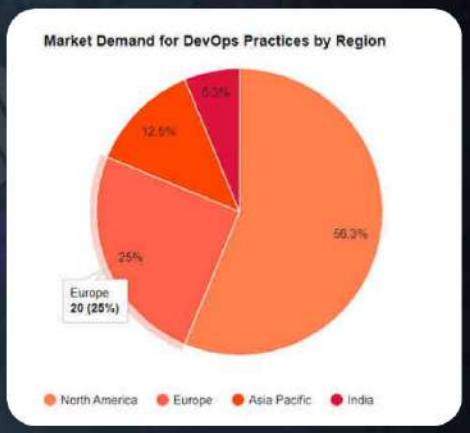
## Introduction to DevOps

"DevOps is the bridge between development and operations"
DevOps is more than just a methodology—it's a cultural shift that emphasizes collaboration between development and operations teams to deliver software faster, more reliably, and with greater efficiency. By integrating automation, monitoring, and continuous feedback loops, DevOps enables organizations to adapt to the rapidly changing tech landscape.
In simple terms, DevOps bridges the gap between developers (who write code) and operations teams (who deploy and manage systems), ensuring a smooth pipeline from idea to production.

## Infographic: The Power of DevOps

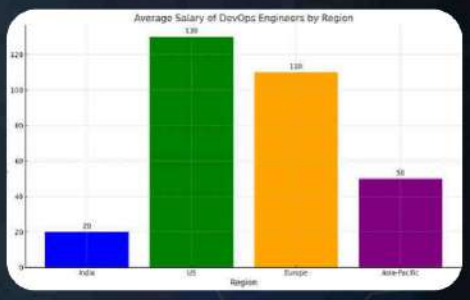
### 1. Market Demand

💡 "80% of Fortune 500 companies use DevOps practices."
This statistic highlights how top organizations rely on DevOps to stay competitive and agile. From technology giants like Google, Amazon, and Netflix to financial powerhouses like JPMorgan and Goldman Sachs, DevOps is at the core of their operations.




### 2. Lucrative Career Opportunities

💰 "The average salary for a DevOps Engineer is ₹15–25 LPA in India and $110,000–$150,000 in the Globally."
With the DevOps skill set being in high demand, professionals in this field are rewarded with competitive salaries and ample growth opportunities.




### 3. Industries Hiring DevOps Professionals

DevOps professionals are not limited to one domain. Here's a glimpse of where they are in demand:

- Technology & IT: Amazon, Google, Microsoft, IBM.
- Finance: JPMorgan Chase, Goldman Sachs, ICICI Bank.
- E-commerce: Flipkart, Walmart, Shopify.
- Healthcare: Cerner, Practo, Medtronic.
- Entertainment: Netflix, Disney+, Spotify.

From startups to multinational corporations, every industry seeks skilled DevOps engineers to streamline processes and enhance product delivery.

TRAIN WITH SHUBHAM

HAPPY LEARNING

# About the Program


## 3 Months Duration

Accelerate your DevOps career in just 3 months with expert-led, hands-on training.


## Weekend Batches

Additional practice and self-check whether you are able to do tasks independently

## Lots of Hands-On Projects

Our program is all about you, lots of hands-on learning with real-world demos to make it real.

## Exercises for each Module

Additional practice and self-check whether you are able to do tasks independently

## Illustrated Handouts

Accompanying handout for each Module. Overview with Key Takeaways

## Active Community

Be part of an exclusive community that share the same journey as yours


## DevOps Certification

Earn a verifiable, recognized digital credential to proof your knowledge

## 24/7 Support

Daily support from experienced engineers, so you don't get stuck

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Bootcamp Features

**Live Features:**

- **Interactive Quizzes** after each module with Leaderboard Tracking.
- Weekly Hackathons to solve real-world problems.
- Phase wise MCQ's Exam

**Internship Program:**

- Work on industry projects under TWS mentorship.

**Job Alerts:**

- Notifications for DevOps job openings.

**Job Opportunities:**

- Tie-ups with hiring partners.
- Regular updates on openings in the DevOps domain.


TRAIN WITH SHUBHAM
HAPPY LEARNING

# About the Program

**Welcome to Your DevOps Learning Journey**

A total of 3 Months Live DevOps Bootcamp is carefully structured into four progressive learning phases, ensuring a comprehensive grasp of the subject, starting from the basics and advancing to job readiness. Each phase is tailored to help you build expertise, confidence, and a portfolio of projects to showcase your skills.

**The Four Phases of Learning**

## Phase 1: Foundation

Master the building blocks of DevOps with a strong foundation

**Linux: Command-line proficiency for server management.**

**Git: Version control for tracking and managing code changes.**


**Shell Scripting: Automate routine tasks with efficient scripts.**

## Phase 2: DevOps Core Tools

Gain hands-on experience with essential tools for DevOps workflows

**Jenkins: Automate CI/CD pipelines.**


**Docker: Containerize applications for seamless deployment.**

**Kubernetes: Orchestrate containers at scale.**

TRAIN WITH SHUBHAM
HAPPY LEARNING

# About the Program

## Phase 3: Advanced Concepts.

**Dive deeper into cutting-edge practices that enhance productivity**


**Ansible:** Ansible automates IT tasks and configurations.

**Terraform:** Manage infrastructure as code.


**Cloud Computing:** Explore AWS for hosting.

**Monitoring Tools:** Use Prometheus and Grafana for real-time insights.

## Phase 4: Career Accelerator

This is where learning meets practical application, preparing you for the job market:

**Internships:** Gain real-world experience by working on live projects.

**Resume Preparation:** Build an ATS-compliant resume highlighting your DevOps skills.

**Mock Interviews:** Practice with industry professionals to ace interviews.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Syllabus Overview and TimeLine

**Briefing**

Induction on overview of the 12-week journey and what you aim to achieve.

1

2

**Weeks 1-3**

**Linux, Git, and Shell Scripting.**

**Weeks 4-6:**

**Docker, Kubernetes, and Jenkins.**

3

4

**Weeks 7-9:**

**Cloud, Terraform, and Monitoring Tools.**

**Weeks 10-12:**

**Real-Time Projects, Mock Interviews, and Job Prep.**

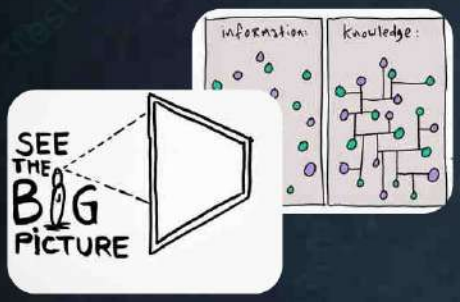
5

TRAIN WITH SHUBHAM

HAPPY LEARNING

# What You'll Learn (Skills)

**By the end of the bootcamp, you can:**




See the Bigger Picture by Connecting all the Dots.


Have Holistic Understanding of DevOps

Master CI/CD pipelines with Jenkins.

Deploy scalable apps with Kubernetes.

Automate infrastructure with Terraform.


Monitor applications using Prometheus and Grafana.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 1

## Linux Essentials

### Module 1: Foundations of Linux and Operating Systems

1. **What is an Operating System?**
   - OS as an abstraction layer between applications and hardware.
   - Tasks of an OS
     - Resource Allocation and Management.
     - File Management.
     - Device Management.
     - Security.
     - Networking.
     - Multi-user and Multitasking in Linux.
2. **Components of an OS**
   - **Kernel:** The core of Linux.
   - **Application Layer:** Interaction through GUIs or CLIs.
   - **How to interact with the kernel:** System calls and commands.
3. **Introduction to the Big 3 Operating Systems**
   - Comparison of Windows, macOS, and Linux.
   - Client OS vs. Server OS.
   - Why Linux is preferred in DevOps.

---

### Module 2: Virtualization and Linux VMs

1. **Understanding Virtualization and Virtual Machines**
   - What is virtualization? (Example: Running multiple "computers" on one physical machine).
   - Architecture: Virtualization layers and hypervisors.
2. **Hypervisors**
   - **Type 1: Bare-Metal Hypervisors (e.g., VMware ESXi, KVM).**
   - **Type 2: Hosted Hypervisors (e.g., VirtualBox, VMware Workstation).**
   - **Differences and use cases.**
3. **Setting up a Linux VM**
   - Hands-on: Installing VirtualBox and creating an Ubuntu VM.
   - Exploring VM settings: Memory, CPUs, and storage.
   - Installing Linux using ISO files.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase I : Foundation
# Week I
# Linux Essentials

## Module 3: Exploring the Linux File System

1. **Linux File System Basics**
   - Everything in Linux is a file: Explanation and examples.
   - File types: Regular files, directories, and binaries.
   - Hidden files: .filename convention.
2. **Understanding the Linux Directory Structure**
   - Important directories: /home, /etc, /var, /bin, /tmp, etc.
   - Navigating the file system: pwd, ls, cd.
   - File permissions and ownership: Basics and umask.
3. **GUI vs. CLI in Linux**
   - Advantages of using CLI in DevOps workflows.

---

## Module 4: Managing Software on Linux

1. **What is a Package Manager?**
   - Purpose and tasks of a package manager.
   - Understanding software packages and repositories.
   - Pre-installed package managers (e.g., APT, YUM).
2. **Hands-On with Package Managers**
   - Basic commands for installing, removing, and updating software using apt and yum.
   - Adding repositories and using PPAs (Personal Package Archives).
3. **Alternative Ways to Install Software**
   - Binary downloads.
   - Source code compilation.

---

## Module 5: Working with Text Editors

1. **Introduction to Nano, Vi, and Vim**
   - What is Vim?
   - Differences and use cases for CLI-based text editors in Linux.
2. **Vim Basics**
   - Modes: Insert and command mode.
   - Editing, saving, and exiting files.
3. **Nano Basics**
   - Basic commands and modes for Nano.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 1
# Linux Essentials

## Module 6: Linux User and Permission Management

1. **Understanding User Accounts**
   - Types of accounts: User, superuser, service.
   - Differences and use cases.
   - /etc/passwd file overview.
   - UID: User Identifier.
2. **Managing Users and Groups**
   - Creating, modifying, and deleting users and groups:
     - High-level utilities: adduser, addgroup.
     - Low-level utilities: useradd, groupadd.
     - Differences and use cases.
   - Sudoers configuration: Managing privileged access.
3. **Understanding Permissions**
   - File ownership: User, group, others.
   - Permission types: Read, write, execute.
   - Changing permissions using symbolic (chmod) and numeric methods.

---

## Module 7: Mastering the Command Line

1. **Essential Commands for File and Directory Management**
   - File operations: touch, rm, mv, cp.
   - Directory operations: mkdir, rmdir.
2. **Pipes and Redirects**
   - Using | for chaining commands.
   - Redirecting output with > and >>.
3. **Practical Examples**
   - Combining commands for log analysis with grep and less.
   - Using wildcards and subshells for advanced use cases.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 2
# Linux Advanced

## Module 8: Introduction to Shell Scripting

1. **Why Shell Scripting?**
   - Automation use cases in DevOps.
   - Writing and executing shell scripts.
2. **Key Shell Scripting Concepts**
   - Variables and conditionals.
   - Built-ins, boolean operators, and loops (for, while).
   - File test operators and relational operators.
   - Passing arguments to a script using positional parameters.
3. **Advanced Shell Scripting**
   - Functions.
   - Debugging scripts: bash -x, set -x.
   - Scheduling scripts with cron.
   - Hands-on projects: Automated backups and log cleanup.

---

## Module 9: Environment Variables

1. **Understanding Environment Variables**
   - What are they? (Analogy: Environment-specific configurations).
   - System-wide vs. user-specific variables.
2. **Working with Environment Variables**
   - Creating and persisting variables.
   - Real-world use cases in application deployment.

---

## Module 10: Networking Essentials

1. **Introduction to Networking**
   - LAN, WAN, MAN, and PAN.
   - Key components: Switches, routers, gateways.
2. **IP Addressing**
   - IP, subnet masks, and gateways.
   - CIDR notation and subnets.
3. **Essential Networking Commands for DevOps**
   - ping, ifconfig, netstat, traceroute.
   - Basics of firewalls (iptables or ufw).

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 2
# Linux Advanced

## Module 11: Secure Shell (SSH)

1. **What is SSH?**
   - Secure remote communication between computers.
   - Use cases in DevOps workflows.
2. **How SSH Works**
   - Authentication methods: Username/password vs. SSH key pairs.
   - Configuring SSH keys for secure access.
3. **Practical Applications of SSH**
   - Connecting to remote servers.
   - Using SSH in automation workflows.
   - Managing servers using ~/.ssh/config.

---

# Phase 1 : Foundation
# Week 3
# Git Fundamentals

## Module 1: Introduction to Version Control

- **What is Version Control?**
  - Version control tracks and manages changes to a project over time, allowing multiple people to work on the same project without interfering with each other's work.
- **Git vs. GitHub:**
  - Git: A tool that helps track and manage your code's history.
  - GitHub: A cloud service where you can store and share your Git repositories.

---

## Module 2: Basic Concepts of Git

- **Repository:** A place where all your project files and their history are stored.
- **Commit:** A snapshot of changes made to files in the repository.
- **Branch:** A separate line of development in the repository.
- **HEAD:** A pointer to the current commit or branch you're working on.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 3

## Git Fundamentals

### Module 3: Setup Git Repository

- **Installing Git:** Get Git from git-scm.com.
- **Configure Git:** Set user name and email globally:

```
git config --global user.name "Name"
git config --global user.email "abc@abc.com"
```

---

### Module 4: Working with Git

- **Initialize Git Locally:**

```
git init  # Initializes a new Git repository in the current directory
```

- **Clone a Repository (copy an existing repo to your local machine):**

```
git clone <repository-url>
```

---

### Module 5: Git Branching and Workflow

- **Create a Branch:**

```
git branch <branch-name>  # Creates a new branch but doesn't switch to it
```

- **Switch to a Branch:**

```
git checkout <branch-name>  # Switch to the specified branch
```

- **Create and Switch to a New Branch in One Command:**

```
git checkout -b <branch-name>  # Create and immediately switch to the new branch
```

- **Delete a Branch:**

```
git branch -d <branch-name>  # Delete a local branch
git push origin --delete <branch-name>  # Delete a branch from the remote repository
```

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 3

# Git Fundamentals

## Module 6: Git Merge and Rebase

**Merge** and **Rebase** are both used to combine changes from different branches, but they handle the history in different ways.

- **Git Merge:** Combines changes from different branches and creates a new merge commit.
  - **Merge Command:**

    ```
    git checkout main  # Switch to the main branch
    git merge feature-xyz  # Merge feature-xyz branch into the main branch
    ```

- **Git Rebase:** Rewrites the history by applying your commits on top of another branch, resulting in a cleaner, linear history.
  - **Rebase Command:**

    ```
    git fetch origin  # Fetch the latest changes from the remote
    git rebase origin/main  # Rebase your branch on top of the latest 'main'
    ```

---

## Module 7: Undoing Changes with Git

- **Git Restore:** Allows you to undo changes to your working directory and staging area.
  - **Restore Modified Files:**

    ```
    git restore <file-name>  # Discards changes in the working directory
    ```

  - **Restore Staged Changes:**

    ```
    git restore --staged <file-name>  # Unstages a file without discarding changes
    ```

- **Git Reset:** Moves the HEAD pointer to a previous commit. This is useful for undoing commits.
  - **Reset to a Previous Commit (soft, mixed, hard):**

    ```
    git reset --soft <commit-hash>  # Keeps changes in the staging area
    git reset --mixed <commit-hash>  # Keeps changes in the working directory
    git reset --hard <commit-hash>  # Discards changes and resets to the commit
    ```

- **Git Revert:** Creates a new commit that undoes the changes introduced by a previous commit.
  - **Revert Command:**

    ```
    git revert <commit-hash>  # Creates a new commit to undo a previous commit
    ```

- **Git Commit Amend:** Allows you to modify the last commit (e.g., fix a typo or add a forgotten file).
  - **Amend Command:**

    ```
    git commit --amend  # Modify the last commit (e.g., change the commit message or add more changes)
    ```

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 1 : Foundation
# Week 4

# Git Advanced

## Module 8: Resolving Merge Conflicts

- **What is a Merge Conflict?**
  - A merge conflict happens when Git can't automatically merge changes because both branches modified the same part of a file.
- **How to Resolve Merge Conflicts:**
  - **Step 1:** Identify the conflict in the file (Git marks it with conflict markers <<<<<<<, =======, >>>>>>>).
  - **Step 2:** Manually edit the file to resolve the conflict.
  - **Step 3:** Stage the resolved file:
    git add <file-name>
  - **Step 4:** Complete the merge:
    git merge --continue  # If you were in the middle of a merge
- **Avoiding Merge Conflicts:**
  - Communicate with teammates about changes.
  - Frequently **pull** changes from the main branch to keep your branch up to date:
    git pull origin main  # Pull latest changes from the main branch

---

## Module 9: Working with SSH Keys for Authentication

- **What is SSH?**
  - SSH keys provide a secure way of logging into your GitHub (or other Git hosts) without using your username and password.
- **Generate SSH Key Pair:**
  - **Step 1:** Generate an SSH key:
    ssh-keygen -t rsa -b 4096 -C "your.email@example.com"  # Generate SSH key pair
    Follow the prompts to save the key.
  - **Step 2:** Add SSH key to your GitHub account:
    - **Copy the public key:**
      cat ~/.ssh/id_rsa.pub  # Display the public key
    - Go to **GitHub -> Settings -> SSH and GPG keys -> New SSH key**, then paste the key.
  - **Step 3:** Test the connection:
    ssh -T git@github.com  # Test the SSH connection with GitHub

# Phase I : Foundation
# Week 4

# Git Advanced

## Module 10: Git Client (GUI or Command Line Tool)

- **Git Command Line:**
  - Git's command line interface (CLI) is powerful and flexible, offering all the advanced features you need.
- **Git GUI Tools:**
  - GUI tools provide a more visual way to interact with Git, making it easier to manage branches, commits, and merges. Popular Git clients include:
    - **GitHub Desktop:** A simple GUI for managing repositories and commits.
    - **SourceTree:** A powerful Git GUI with support for Git and Mercurial repositories.
    - **GitKraken:** A modern Git client with a beautiful interface and built-in support for Git flow.

---

## Module 11: Additional Advanced Git Commands

- **Git Cherry-Pick:** Apply a specific commit from one branch to another.
  git cherry-pick <commit-hash>

- **Git Diff:** Show the differences between two commits or between a commit and the working directory.
  git diff <commit1> <commit2> # Show changes between two commits

- **Git Log:** View the commit history of the repository.
  git log # View a simple log of commits
  git log –oneline # View logs in a compact format

- **Git Tag:** Mark specific commits with labels (useful for releases).
  git tag <tag-name> # Create a tag at the current commit

---

## Module 12: Git Stash and Git Pop

- **Git Stash:** Temporarily save changes without committing.
  git stash # Save changes to stash
  git stash list # View stashed changes

- **Git Pop:** Retrieve changes from the stash.
  git stash pop # Apply the last stashed changes
  git stash apply <stash-name> # Apply a specific stash

# Phase 1 : Foundation
# Week 4

## Git Advanced

### Module 13: Advanced Merging Strategies (Squash Merge)

- **Squash Merging:** Combine multiple commits into a single commit to keep a clean history.

```
git merge --squash <branch-name>
git commit  # Commit the squashed changes
```

---

### Module 14: Git Hooks (Advanced)

- Automating tasks with **Git Hooks** (e.g., pre-commit, pre-push).
  - Example: Pre-push hook to run tests before pushing:

```
git commit --pre-commit-hook  # Check code quality or run tests before commit
```

---

### Module 15: Rebasing and Interactive Rebasing

- **Interactive Rebase:** Clean up commit history by squashing or rewording commits.

```
git rebase -i HEAD~3  # Interactively rebase the last 3 commits
```

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 5

## Docker Fundamentals

### Module 1: Introduction to Build and Package Manager Tools

**1.1 What are Build and Package Manager Tools?**

- Overview of build and package manager tools in DevOps
- Importance of automating build processes
- Examples: Maven, Gradle, npm, pip, etc.

---

**1.2 What is an "Artifact"?**

- Definition and types of artifacts (JAR, WAR, Docker images, etc.)
- Role of artifacts in deployment and distribution

---

**1.3 What Does "Building the Code" Mean?**

- Definition of the build process (compiling, packaging, testing)
- Continuous Integration (CI) and Continuous Deployment (CD) in DevOps

---

**1.4 What is an "Artifact Repository"?**

- Introduction to artifact repositories (e.g., Nexus, Artifactory)
- Managing and storing versioned artifacts
- Benefits of centralized artifact storage

---

**1.5 Different Build Tools for Different Programming Languages**

- **Java:** Maven, Gradle
- **JavaScript:** npm, Yarn, Webpack
- **Python:** pip, virtualenv
- **C/C++:** Conan
- **C#:** NuGet
- **Golang:** dep
- **Ruby:** RubyGems

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 5

## Docker Fundamentals

### 1.6 Building Artifacts for Specific Projects

- **Java (Gradle and Maven):** Building artifacts using Gradle and Maven
- **JavaScript (npm/Yarn):** Creating build scripts for front-end applications
- **Python:** Building Python packages
- **C/C++:** Using Conan for C/C++ builds
- **Golang:** Building Go applications with dep

---

### 1.7 Build Tools for Software Development

- Managing dependencies
- Integrating build tools with version control systems (e.g., Git)

---

### 1.8 Build JavaScript Applications

- Setting up build scripts for modern JavaScript apps
- Integrating with bundlers (Webpack)
- Managing front-end dependencies with npm/yarn

---

### Module 2: Introduction to Docker and Containers

### 2.1 What is Docker?

- Overview of Docker as a containerization platform
- Docker's role in modern software development and DevOps

---

### 2.2 What is a Container?

- Definition and key differences between containers and traditional virtual machines
- Containerization and its benefits in scalability, portability, and efficiency

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 5

## Docker Fundamentals

**2.3 Container vs Image**

- Difference between a container and a Docker image
- Understanding the life cycle of a container and image

---

**2.4 Docker vs Virtual Machines**

- Comparing Docker containers and Virtual Machines (VMs)
- Advantages of Docker over VMs

---

**2.5 Most Popular Container Technologies**

- Overview of popular container technologies: Docker, Containerd, and CRI-O

---

**2.6 Docker Architecture & Components**

- **Docker Daemon:** Manages Docker containers and images
- **Docker Client:** Interacts with Docker Daemon
- **Docker Registry:** Stores Docker images (public and private repositories)
- **Docker Images:** Immutable snapshots of a container
- **Docker Containers:** Running instances of Docker images
- **Dockerfile:** Defines the instructions to build Docker images

---

**2.7 Docker Installation and Setup**

- Installing Docker on Linux, Windows, and macOS
- Introduction to Docker Desktop for Windows and macOS

---

# Docker Fundamentals

## Module 3: Docker Commands and Usage

### 3.1 Main Docker Commands

- **docker run:** Create and start a container from an image
- **docker pull:** Pull images from Docker registry
- **docker start/stop:** Start or stop a container
- **docker ps:** View running containers
- **docker ps -a:** Show all containers (running and exited)
- **docker images:** List available images

---

### 3.2 Debug Commands

- **docker logs:** Fetch logs of a container
- **docker exec -it:** Start a bash session inside a container for debugging
- **docker inspect:** View detailed information about containers and images

---

### 3.3 Docker Ports

- Mapping container ports to host ports
- Exposing ports for communication between containers and the outside world

---

## Module 4: Dockerfile and Docker Image Management

### 4.1 Dockerfile Basics

- Writing a basic Dockerfile to define how a Docker image is built
- **FROM:** Specify the base image
- **COPY:** Copy files to the container
- **ENV:** Set environment variables
- **ENTRYPOINT:** Define the default executable for the container
- **RUN:** Execute commands during image build
- **CMD:** Default command executed when the container starts

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 6

## Docker Advanced

**4.2 Multi-Stage Builds in Dockerfile**

- Benefits of multi-stage builds for reducing image size
- Writing multi-stage Dockerfiles
- Use cases for multi-stage builds

---

**4.3 Distroless Images**

- What are distroless images and when to use them
- Benefits of using distroless images for security and reduced attack surface

---

**4.4 Real-World Dockerfile Examples**

- Example 1: Building a Java application with Gradle/Maven
- Example 2: Building a Node.js application with npm
- Example 3: Building a Python application with pip

---

**4.5 Tagging and Pushing Docker Images**

- How to tag Docker images using docker tag
- Pushing images to Docker Hub or private repositories

---

**Module 5: Docker Compose and Multi-Container Applications**

**5.1 What is Docker Compose?**

- Overview of Docker Compose for managing multi-container applications
- Syntax and structure of a docker-compose.yml file

---

**5.2 Docker Compose - Running Multiple Services**

- Defining services, networks, and volumes in a Compose file
- Example project: A multi-container app with a web server and database

---

**5.3 Managing Containers with Docker Compose**

- Starting, stopping, and scaling services with Docker Compose
- View logs and inspect containers

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 6

## Docker Advanced

### Module 6: Docker Volumes and Data Persistence

**6.1 Docker Volumes: Why Persistence is Important**

- What are Docker volumes and when to use them
- Differences between host volumes, anonymous volumes, and named volumes

---

**6.2 How Docker Volumes Work**

- Creating and mounting Docker volumes
- Example use case: Persisting database data across container restarts

---

**6.3 Configuring Volumes in Docker Compose**

- Defining volumes in docker-compose.yml
- Managing shared volumes between containers

---

### Module 7: Docker Networking Deep Dive

**7.1 Bridge Network (Default)**

- **Use Case:** Connecting containers on the same host.
- **Working with Docker Compose:** Using bridge networks for multi-container apps.

---

**7.2 Host Network**

- **Use Case:** Removing isolation between Docker containers and the Docker host.

---

**7.3 IPvlan and Macvlan Networks**

- **IPvlan:** Providing custom IP management for containers, with full control over IP address allocation.
- **Macvlan:** Assigning MAC addresses to containers for direct network access on a physical network.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 6

## Docker Advanced

### Module 8: Advanced Docker Concepts and Real-World Scenarios

**8.1 Private Docker Registry**

- Setting up a private Docker registry using AWS or Nexus
- Pushing and pulling images from private repositories

---

**8.2 CI/CD Integration with Docker**

- Using Docker in continuous integration and deployment pipelines
- Automating the build, test, and deployment of containerized applications

---

**8.3 Docker with Nexus Repository**

- Pushing and pulling Docker images to/from Nexus repository
- Running Nexus as a Docker container

---

**8.4 Real-World Scenario: Deploying Containerized Applications**

- Deploying a full-stack application using Docker and Docker Compose
- Example: A Node.js backend with a MongoDB database

---

### Module 9: Final Project

**9.1 Demo Project Overview**

- Setting up a complete DevOps pipeline using Docker
- Building, testing, and deploying a multi-container application with Docker Compose
- Pushing Docker images to a private Docker registry

---

**9.2 Hands-On Project Development**

- Developing and deploying a containerized JavaScript or Java application
- Using Docker volumes to manage data persistence
- Automating the process with Docker Compose and CI/CD tools

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 6

## Docker Advanced

### Module 8: Advanced Docker Concepts and Real-World Scenarios

**8.1 Private Docker Registry**

- Setting up a private Docker registry using AWS or Nexus
- Pushing and pulling images from private repositories

---

**8.2 CI/CD Integration with Docker**

- Using Docker in continuous integration and deployment pipelines
- Automating the build, test, and deployment of containerized applications

---

**8.3 Docker with Nexus Repository**

- Pushing and pulling Docker images to/from Nexus repository
- Running Nexus as a Docker container

---

**8.4 Real-World Scenario: Deploying Containerized Applications**

- Deploying a full-stack application using Docker and Docker Compose
- Example: A Node.js backend with a MongoDB database

---

### Module 9: Final Project

**9.1 Demo Project Overview**

- Setting up a complete DevOps pipeline using Docker
- Building, testing, and deploying a multi-container application with Docker Compose
- Pushing Docker images to a private Docker registry

---

**9.2 Hands-On Project Development**

- Developing and deploying a containerized JavaScript or Java application
- Using Docker volumes to manage data persistence
- Automating the process with Docker Compose and CI/CD tools

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 7

## Jenkins Fundamenatals

### Module 1: Introduction to Build Automation and CI/CD

- **What is Build Automation?**
  - Automating source code retrieval.
  - Executing automated tests.
  - Compiling code/building Docker images.
  - Pushing artifacts to repositories.
  - Deploying artifacts to environments.
  - Role of Build Automation in CI/CD and DevOps.
- **What is CI/CD?**
  - **Continuous Integration:** Merging, testing, and building code changes in real-time.
  - **Continuous Deployment:** Automatically deploying validated builds to various environments.
  - Benefits and importance of CI/CD pipelines in modern software delivery.
- **Comparing Build Automation Tools:**
  - Overview of Jenkins, GitLab, Travis CI, Bamboo, and TeamCity.

---

### Module 2: Getting Started with Jenkins

- **Introduction to Jenkins:**
  - Overview of Jenkins.
  - Jenkins' role in DevOps pipelines.
  - Key features and integrations.
- **Jenkins Architecture:**
  - Master-agent model.
  - How Jenkins manages tasks and workloads.
- **Installing Jenkins:**
  - Running Jenkins as a Docker container.
  - Installing Jenkins directly on Linux/Windows/MacOS.
- **Jenkins User Roles:**
  - Administrator roles (plugin management, setup, etc.).
  - User roles (job creation, pipeline management, etc.).

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 7

## Jenkins Fundamenatals

### Module 4: Advanced Pipeline Configurations

- **Pipeline as Code:**
  - Storing Jenkinsfiles in version control.
  - Advantages of declarative pipelines over scripted pipelines.
- **Creating Robust Pipelines:**
  - Parallel execution of stages.
  - Dynamically triggering stages and jobs.
  - Integrating external Groovy scripts.
- **Using Shared Libraries:**
  - Creating and managing shared libraries.
  - Configuring shared libraries globally and per project.
  - Reusing code across pipelines.

---

### Module 5: Integrating Jenkins with DevOps Tools

- **Source Code Management (SCM) Integration:**
  - Configuring GitHub, GitLab, Bitbucket repositories in Jenkins.
  - Setting up Webhooks for automatic triggers.
- **Build Automation Tool Integration:**
  - Configuring Maven, Gradle, npm, and Docker in Jenkins.
- **Deployment Automation:**
  - Deploying artifacts to AWS, Azure, Kubernetes, and Docker Swarm.
  - Automating rollbacks and blue-green deployments.
- **Test Automation Integration:**
  - Configuring Jenkins with Selenium, JUnit, and TestNG.
  - Managing test reports and logs.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 8

## Jenkins Advanced

### Module 6: Credentials and Security Management

- **Managing Credentials in Jenkins:**
  - Credential scopes (System vs. Global).
  - Credential types (Secret text, username/password, SSH keys, Docker host certificates, etc.).
- **Securing Jenkins:**
  - Role-based access control.
  - Securing Jenkinsfile parameters.
  - Backup and restore strategies.

---

### Module 7: Monitoring and Optimizing Jenkins

- **Monitoring Jenkins Performance:**
  - Using plugins for performance monitoring.
  - Tracking build durations and resource usage.
- **Optimizing Jenkins Jobs:**
  - Leveraging distributed builds.
  - Managing and cleaning up workspace and build history.

---

### Module 8: Versioning and Best Practices

- **Software Versioning:**
  - Understanding semantic versioning (major, minor, patch).
  - Automating dynamic versioning in builds.
- **Jenkins Best Practices:**
  - Using pipeline as code.
  - Storing Jenkinsfiles in repositories.
  - Creating modular and reusable pipeline scripts.
  - Using shared libraries effectively.
  - Regularly updating Jenkins and plugins.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 8

## Jenkins Advanced

### Module 9: Real-World Use Cases and Hands-On Labs

- **End-to-End CI/CD Pipeline:**
  - Building a multi-stage pipeline for a Java application.
  - Containerizing with Docker and deploying to Kubernetes.
- **Complex Pipeline Scenarios:**
  - Multibranch pipelines for microservices.
  - Automating workflows for large-scale deployments.
- **Troubleshooting Jenkins:**
  - Common pipeline errors and their solutions.
  - Debugging failed builds and logs.

---

### Module 10: Emerging Trends and Future of Jenkins

- **Scaling Jenkins:**
  - Setting up Jenkins in a Kubernetes cluster.
  - Using Jenkins X for cloud-native CI/CD.
- **Jenkins Alternatives:**
  - Exploring modern alternatives and integrations.
  - When to choose Jenkins over other tools.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 9

# Kubernetes Fundamentals

## Module 1: Introduction to Containers and Kubernetes

1. **Understanding Monoliths and Microservices**
   - What are Monoliths?
   - Transition to Microservices architecture
   - Benefits and challenges of Microservices
2. **The Evolution from VMs to Containers**
   - Differences between VMs and Containers
   - Why containers became a game-changer
3. **Container Fundamentals**
   - Containers as Linux processes
   - Overview of Linux namespaces (demo: isolated processes)
   - Introduction to Linux cgroups
4. **Introduction to Kubernetes**
   - What is Kubernetes?
   - Why Kubernetes? The need for orchestration
   - Kubernetes vs traditional container management
   - Core features of Kubernetes

---

## Module 2: Kubernetes Architecture and Components

1. **Core Components**
   - Control Plane (API Server, Scheduler, Controller Manager, etcd)
   - Node Components (kubelet, kube-proxy, container runtime)
   - Addons (DNS, CNI plugins, etc.)
2. **Kubeconfig File**
   - Purpose of the Kubeconfig file
   - Structure and usage
   - Creating a new user with a dedicated Kubeconfig file
3. **Interacting with Kubernetes API**
   - GVR (Group, Version, Resource) and GVK (Group, Version, Kind)
   - Using curl to query Kubernetes API
   - Using kubectl proxy to interact with the API
   - Difference between REST API and CLI usage

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 9

# Kubernetes Fundamentals

## Module 3: Workloads and YAML Basics

1. **Declarative vs Imperative**
   - Pros and cons of both approaches
   - Hands-on examples of kubectl run (imperative) vs YAML files (declarative)
2. **YAML Basics**
   - Syntax and structure
   - Key-value pairs, lists, and anchors
   - Practice creating YAML manifests
3. **Pods**
   - What is a Pod?
   - Pod lifecycle and phases
   - Init containers (use cases and examples)
   - Sidecar containers (use cases like logging, proxy, etc.)

---

## Module 4: Advanced Pod Concepts

1. **Sidecar Container Example**
   - Real-world example (logging, service mesh, or caching)
2. **Pod Disruption Budgets**
   - Purpose and configuration
   - Use cases in production environments
3. **Resource Requests and Limits**
   - Quality of Service (QoS) tiers
   - Understanding resource starvation and overcommitment
4. **Pause Container**
   - Purpose and role in Pod lifecycle
5. **Downward API**
   - Accessing Pod metadata from inside containers

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools

## Week 9

# Kubernetes Fundamentals

### Module 5: Kubernetes Scheduling

1. **Namespaces**
   - Isolation and resource organization
   - Resource quotas and limits in namespaces
2. **Labels and Selectors**
   - Organizing and selecting resources
3. **Node Affinity**
   - Scheduling Pods based on node attributes
4. **Taints and Tolerations**
   - Restricting Pod placement
5. **Pod Priority and Preemption**
   - Priority classes and use cases
6. **Topology Spread Constraints**
   - Ensuring Pod distribution
7. **Scheduling Strategies**
   - Manual binding
   - Pod readiness and its impact on scheduling

---

## Week 10

# Kubernetes Advanced

### Module 6: ReplicaSets, Deployments, and Probes

1. **ReplicaSet**
   - Role and purpose
   - Deletion mechanisms and cascading deletion
2. **Deployments**
   - Rolling updates and rollback strategies
   - Deployment strategies: Recreate, Rolling, Canary
3. **Kubernetes Probes**
   - Liveness, Readiness, and Startup probes
   - Configuration and troubleshooting

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 2 : DevOps Core Tools
Week 10

# Kubernetes Advanced

## Module 7: ConfigMaps and Secrets

1. **ConfigMaps**
   - Using as environment variables
   - Using as volumes (demo)
   - Accessing programmatically from applications
2. **Secrets**
   - Types of secrets (Opaque, TLS, etc.)
   - ImagePullSecrets demo
   - Encrypting secrets

---

## Module 8: StatefulSets, Services, and Ingress

1. **StatefulSets**
   - Stateful application management
   - Headless services with StatefulSets (demo)
2. **Services**
   - ClusterIP, NodePort, and ExternalName
   - Configuring DNS with external services
3. **Ingress**
   - Ingress controllers (NGINX demo)
   - Configuring routes with TLS (Cert-Manager integration)

---

## Module 9: Kubernetes Authentication, Authorization, and Admission

1. **Authentication, Authorization, and Admission (AAA)**
   - Service accounts and default tokens
   - Role-Based Access Control (RBAC)
2. **Admission Controllers**
   - Validating and mutating admission webhooks
   - ImagePolicy webhook demo

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 2 : DevOps Core Tools
# Week 10

# Kubernetes Advanced

## Module 10: Kubernetes Volumes

1. **Volume Concepts**
   - Types of volumes: emptyDir, hostPath, local, etc.
2. **Persistent Volumes (PV) and Persistent Volume Claims (PVC)**
   - Dynamic provisioning with storage classes
   - NFS server-backed PV and PVC (demo)

---

## Module 11: Project Deployment with Kubernetes

1. **End-to-End Deployment**
   - Python Cloud-Native App deployment
   - PostgreSQL via CloudNativePG
   - Ingress and Cert-Manager integration for HTTPS
   - Adding DNS records for production readiness

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 3 : DevOps Core Tools
# Week 11

# Ansible

## Module 1: Introduction to Ansible

- **Overview of Ansible:**
  - Open-source origins: History, vision, and evolution of Ansible.
  - Why Ansible? Challenges it addresses in configuration management and orchestration.
  - Comparison with traditional tools (e.g., Puppet, Chef, SaltStack).
- **Core Features of Ansible:**
  - Agentless architecture.
  - Declarative versus procedural approaches.
  - YAML-based configuration.
  - SSH as a transport mechanism.
- **Industry Use Cases:**
  - Managing configurations across multiple servers.
  - Orchestrating application deployments.
  - Automating cloud provisioning.

---

## Module 2: Getting Started with Ansible

- **Installing Ansible:**
  - **Installation on Linux, macOS, and Windows.**
  - **Setting up Ansible in a virtual environment.**
- **Ansible Configuration Basics:**
  - **Understanding the ansible.cfg file.**
  - **Inventory files: Static and dynamic inventories.**
- **Understanding SSH Keys:**
  - **Setting up password-less SSH access.**
  - **Troubleshooting SSH connection issues.**

---

# Phase 3 : DevOps Core Tools
# Week 11

# Ansible

## Module 3: Ansible Architecture and Workflow

- **Core Components:**
  - Control node.
  - Managed nodes.
  - Modules, plugins, and inventory.
- **Workflow:**
  - Write -> Test -> Execute.
  - Idempotency in Ansible operations.
- **Execution Flow:**
  - Task execution flow.
  - Parallelism and forks in Ansible.

---

## 4. Ansible Ad-Hoc Commands

- **Introduction to Ad-Hoc Commands:**
  - Running quick tasks without playbooks.
- **Examples:**
  - File management.
  - Service control.
  - User management.

---

## 5. Ansible Playbooks

- **Anatomy of a Playbook:**
  - Structure and syntax.
  - Tasks, plays, and handlers.
  - Tags and conditionals.
- **Writing Effective Playbooks:**
  - Reusing code with roles.
  - Using variables and templates.
  - Handling errors and retries.

# Phase 3 : DevOps Core Tools
# Week 11

# Ansible

## Module 6: Ansible Modules

- Core Modules Overview:
  - File, user, service, and package modules.
- Exploring Cloud Modules:
  - AWS, Azure, and GCP modules.
- Writing Custom Modules:
  - Python basics for creating Ansible modules.
  - Best practices for module development.

---

## Module 7: Variables in Ansible

- Types of Variables:
  - Host and group variables.
  - Facts and registered variables.
- Managing Variables:
  - Variable precedence and scope.
  - Encrypting variables using Ansible Vault.

---

## Module 8: Advanced Templating with Jinja2

- Introduction to Jinja2:
  - Syntax and filters.
  - Conditional statements and loops.
- Dynamic Inventory and Configuration:
  - Using Jinja2 templates for dynamic file generation.

---

# Phase 3 : DevOps Core Tools
# Week 11

# Ansible

## Module 9: Roles in Ansible

- **Introduction to Roles:**
  - Why roles? Modular and reusable configurations.
  - Structure of a role.
- **Creating and Using Roles:**
  - Role dependencies and defaults.
  - Sharing roles via Ansible Galaxy.

---

## Module 10: Ansible Galaxy and Collections

- **Introduction to Ansible Galaxy:**
  - Discovering and downloading community roles.
- **Collections in Ansible:**
  - Understanding and using collections.
  - Managing dependencies with collections.

---

## Module 11: Ansible for Cloud Automation

- **Provisioning in AWS, Azure, and GCP:**
  - Writing playbooks for cloud infrastructure automation.
  - Managing multi-cloud environments.
- **Integrating Ansible with Terraform for hybrid environments.**

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 3 : DevOps Core Tools
# Week 11

# Ansible

## Module 12: Ansible Tower / AWX

- Overview:
  - What is Ansible Tower/AWX?
  - Differences between Ansible CLI and Tower.
- Features:
  - Role-based access control (RBAC).
  - Job scheduling and workflows.
  - Visualizing playbook execution.
- Installing and Configuring AWX:
  - Docker-based installation.
  - Managing projects and templates.

---

## Module 13: Security and Compliance with Ansible

- Securing Ansible Operations:
  - Using Ansible Vault to encrypt sensitive data.
  - Role-based access and SSH key management.
- Compliance Automation:
  - Writing compliance playbooks.
  - Auditing and remediation with Ansible.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : DevOps Core Tools
Week 12

# Terraform Fundamentals

## Module 1: Introduction to Terraform

- **What is Terraform?**
  - Definition, origin, and evolution of Terraform.
  - Key features: declarative vs imperative approach.
  - Comparison with other Infrastructure as Code (IaC) tools (e.g., CloudFormation, Ansible).
  - Why Terraform? (Problem-solving capabilities and unique features).
- **Open Source and Beyond**
  - Terraform OSS vs Terraform Cloud vs Terraform Enterprise.
  - Community contributions and support.
- **Infrastructure as Code (IaC): A Paradigm Shift**
  - Why IaC is essential in DevOps.
  - Declarative vs Imperative IaC (examples and use cases).

---

## Module 2: Terraform Fundamentals

- **Core Concepts**
  - Providers, Resources, and State Files.
  - The write-plan-apply workflow in Terraform.
  - Configuration language and its syntax (HCL).
- **Terraform Workflow**
  - Writing basic configuration.
  - Planning infrastructure changes.
  - Applying and managing the state.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : DevOps Core Tools
Week 12

# Terraform Fundamentals

## Module 3: Terraform Architecture

- **Terraform Core**
  - Role in interacting with providers and resources.
  - Execution plans and dependency graph.
- **Providers**
  - What are providers?
  - Built-in vs community providers.
  - Writing custom providers.
- **State Management**
  - Local vs remote state.
  - State locking and consistency.
  - Resolving state drift.
- **Comparison with Other Tools**
  - Terraform vs Ansible: Declarative vs Procedural IaC.
  - Terraform vs CloudFormation: Portability and multi-cloud support.
  - Use case-based comparisons for managing large-scale infrastructure.

---

## Module 4: Terraform CLI and Commands

- **Core Terraform Commands**
  - init, plan, apply, destroy, refresh, validate, fmt.
  - Detailed walkthrough with use cases.
- **Advanced Commands**
  - taint, import, graph, and state manipulation.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : DevOps Core Tools
Week 12

# Terraform Fundamentals

## Module 5: Terraform Variables and Expressions

- **Variables Overview**
  - Input variables, output variables, and locals.
  - Defining and using variables in configurations.
  - Variable precedence and overrides.
- **Advanced Usage**
  - Conditional expressions, dynamic blocks, and loops.
  - Variable validation and debugging.
  - Secrets management with environment variables.

---

## Module 6: State Management and Backends

- **State File Overview**
  - Role of state files in infrastructure management.
  - Secure state management practices.
- **Remote Backends**
  - Configuring remote backends (e.g., S3, Azure Blob, GCS).
  - State locking with DynamoDB or Consul.
  - Real-world scenarios for multi-team collaboration.

---

## Module 7: Terraform Provisioners

- **Understanding Provisioners**
  - Use cases for provisioners (creation vs destruction).
  - Examples: file, local-exec, and remote-exec.
- **Advanced Provisioning**
  - Using user_data with cloud instances.
  - Error handling and failure recovery.

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 3 : DevOps Core Tools
# Week 12

## Terraform Fundamentals

### Module 8: Modularizing Terraform Configurations

- **Introduction to Modules**
  - Importance of reusable code.
  - Using existing modules from Terraform Registry.
- **Creating and Managing Custom Modules**
  - Module structure, best practices, and outputs.
  - Nested modules for complex infrastructures.

---

### Module 9: Workspaces and Environment Management

- **Workspaces Overview**
  - Creating and switching workspaces.
  - Using workspaces for managing multiple environments (dev, staging, production).

---

### Module 10: Multi-Cloud Strategy with Terraform

- **Introduction to Multi-Cloud**
  - Benefits and challenges of multi-cloud adoption.
  - Role of Terraform in multi-cloud infrastructure management.
- **Multi-Cloud Use Cases**
  - Hybrid cloud deployments with AWS, Azure, and GCP.
  - High availability across clouds.
- **Hands-On Project**
  - Provisioning and managing hybrid cloud infrastructure using Terraform.
  - Detailed diagram to demonstrate network interconnectivity, load balancing, and failover setups.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 3 : DevOps Core Tools
# Week 12

## Terraform Fundamentals

### Module 8: Modularizing Terraform Configurations

- **Introduction to Modules**
  - Importance of reusable code.
  - Using existing modules from Terraform Registry.
- **Creating and Managing Custom Modules**
  - Module structure, best practices, and outputs.
  - Nested modules for complex infrastructures.

---

### Module 9: Workspaces and Environment Management

- **Workspaces Overview**
  - Creating and switching workspaces.
  - Using workspaces for managing multiple environments (dev, staging, production).

---

# Phase 3 : Week 13

## Terraform Advanced

### Module 10: Multi-Cloud Strategy with Terraform

- **Introduction to Multi-Cloud**
  - Benefits and challenges of multi-cloud adoption.
  - Role of Terraform in multi-cloud infrastructure management.
- **Multi-Cloud Use Cases**
  - Hybrid cloud deployments with AWS, Azure, and GCP.
  - High availability across clouds.
- **Hands-On Project**
  - Provisioning and managing hybrid cloud infrastructure using Terraform.
  - Detailed diagram to demonstrate network interconnectivity, load balancing, and failover setups.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

# Phase 3 : Advanced Concepts
# Week 13

# Terraform Advanced

## Module 11: Automation with Terraform

- **Automating Cloud Infrastructure**
  - Provisioning an EKS cluster with Terraform.
  - Real-world multi-cloud automation scenarios.
- **CI/CD Integration**
  - Using Terraform with Jenkins, GitHub Actions, and GitLab CI.
  - Testing Terraform configurations with Terratest.

---

## Module 12: Secrets Management and HashiCorp Vault

- **Securing Sensitive Data**
  - Managing secrets in Terraform configurations.
  - Integrating with HashiCorp Vault for dynamic secrets.

---

## Module 13: Advanced Topics

- **Custom Terraform Providers**
  - Building and testing a provider from scratch.
- **Performance Optimization**
  - Resource dependencies and parallel execution.
  - Tips for large-scale infrastructure management.
- **Cost Estimation and Tracking**
  - Terraform Cloud Cost Estimation features.
  - Using third-party tools for cost tracking.

---

# Phase 3 : Advanced Concepts
# Week 13

# Terraform Advanced

## Module 14: Real-World Project-Based Modules

- **Project 1: Deploying a Scalable Web Application**
  - Use Terraform to deploy a load-balanced, auto-scaled web app on AWS.
- **Project 2: Multi-Cloud Deployment**
  - Provision infrastructure across AWS and GCP using Terraform.

---

## For Mega Projects

- **Project 3: Kubernetes Cluster Management**
  - Deploy and manage an EKS or AKS cluster with Terraform.
- **Project 4: Enterprise-Grade Monitoring and Logging**
  - Integrate Terraform with Prometheus and Grafana for infrastructure monitoring.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : Advanced Concepts
Week 14

# AWS Cloud Computing

## Module 1: Introduction to Cloud Computing and AWS

1. **Understanding Cloud Computing**
   - What is Cloud Computing?
   - Advantages of Cloud Computing (cost, scalability, availability).
2. **Types of Clouds**
   - Public Cloud (e.g., AWS, Azure, Google Cloud).
   - Private Cloud (e.g., on-premise solutions).
   - Hybrid Cloud.
   - Comparison between Public, Private, and Hybrid Cloud.
3. **Introduction to AWS**
   - What is AWS?
   - AWS Global Infrastructure: Regions, Availability Zones, Edge Locations.
   - Core AWS Services Overview.

---

## Module 2: Identity and Access Management (IAM)

1. **IAM Basics**
   - What is IAM and its purpose?
   - Users, Groups, Roles, and Policies.
2. **Hands-On**
   - Creating IAM Users, Groups, and Roles.
   - Attaching Policies to Control Access.
3. **Best Practices**
   - Principle of Least Privilege.
   - Enforcing MFA (Multi-Factor Authentication).
   - Auditing IAM Activity (CloudTrail).

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : Advanced Concepts
Week 14

# AWS Cloud Computing

## Module 3: Compute with EC2

1. **Introduction to EC2**
   - What is EC2?
   - Types of EC2 Instances (General Purpose, Compute Optimized, etc.).
   - Choosing the Right Instance Type.
2. **Launching EC2 Instances**
   - Launch an Instance and Connect via SSH (Windows, Linux, macOS).
   - Configure Security Groups.
   - Managing EC2 Instances (Start, Stop, Reboot, Terminate).
3. **Deploying Applications**
   - Install and Configure Jenkins on EC2.
   - Deploy a Sample Application.

---

## Module 4: Networking with AWS VPC

1. **Virtual Private Cloud (VPC)**
   - Understanding VPC: Isolation and Security.
   - Subnets: Public vs. Private Subnets.
   - IP Addressing and CIDR Blocks.
2. **VPC Components**
   - Route Tables, Internet Gateways, NAT Gateways.
   - Network ACLs (Access Control Lists).
   - Security Groups.
3. **Advanced Networking**
   - VPC Peering.
   - Traffic Mirroring.
   - VPC Flow Logs.
   - VPN Connections.

---

# AWS Cloud Computing

## Module 5: AWS Storage (S3 and Beyond)

1. **Introduction to S3**
   - What is S3?
   - Key Features: Durability, Scalability, Object Storage.
   - Uploading and Downloading Objects.
2. **Advanced S3 Features**
   - Versioning and Lifecycle Policies.
   - Cross-Region Replication.
   - Event Notifications and Triggers.
   - S3 Encryption (At Rest and In-Transit).
   - S3 SDKs and APIs.
3. **Other AWS Storage Services**
   - EBS (Elastic Block Store).
   - EFS (Elastic File System).

---

## Module 6: AWS Networking and Route 53

1. **Introduction to Route 53**
   - DNS Basics and Domain Registration.
   - Setting Up DNS Records (A, CNAME, MX).
2. **Advanced DNS Features**
   - Health Checks.
   - Routing Policies (Simple, Weighted, Latency-Based, Failover).
   - DNS-Based Load Balancing.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : Advanced Concepts
Week 15

# AWS Cloud Computing

## Module 7: Monitoring and Logging

1. **AWS CloudWatch**
   - **Metrics, Logs, and Alarms.**
   - **Real-World Scenarios: Monitoring EC2 Instances and Applications.**
2. **AWS CloudTrail**
   - **Logging API Calls.**
   - **Security and Compliance Use Cases.**
3. **AWS Config**
   - **Resource Inventory and Configuration History.**
   - **Compliance Management.**

---

## Module 8: Serverless Computing with AWS Lambda

1. **Introduction to Serverless**
   - What is Serverless Computing?
   - Benefits of AWS Lambda (Auto-Scaling, Pay-as-You-Go).
2. **Hands-On with Lambda**
   - Writing and Deploying Lambda Functions.
   - Integrating Lambda with S3, DynamoDB, and API Gateway.
3. **Real-World Use Cases**
   - Automated File Processing.
   - Event-Driven Applications.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : Advanced Concepts
Week 15

# AWS Cloud Computing

## Module 9: Infrastructure as Code (IaC) with AWS CloudFormation

1. **Introduction to CloudFormation**
   - What is IaC?
   - Benefits of Using CloudFormation.
2. **Hands-On**
   - Writing CloudFormation Templates.
   - Managing Stacks.
   - Automating Infrastructure Provisioning.

---

## Module 10: AWS Developer Tools

1. **CodeCommit**
   - Setting Up Repositories.
   - Collaborating on Code with Teams.
2. **CodePipeline**
   - Configuring CI/CD Pipelines.
   - Automating Build, Test, and Deploy Workflows.
3. **CodeBuild**
   - Defining Build Specifications.
   - Integrating with Other AWS Services.
4. **CodeDeploy**
   - Automating Application Deployment.
   - Rolling Back Deployments.

---

TRAIN WITH SHUBHAM
HAPPY LEARNING

Phase 3 : Advanced Concepts
Week 15

# AWS Cloud Computing

## Module 11: Advanced Services and Topics

- **Containerization and Orchestration**
  - AWS Elastic Container Registry (ECR): Managing Docker Images.
  - AWS Elastic Kubernetes Service (EKS): Deploying and Scaling Kubernetes Clusters.
- **AWS Elastic Load Balancer**
  - Types of Load Balancers (Application, Network, Classic).
  - Configuring and Using Load Balancers.
- **CloudFront (CDN)**
  - Content Delivery Networks and Use Cases.
  - Setting Up CloudFr/ont Distributions.
- **AWS Systems Manager**
  - Centralized Management of Resources.
  - Automating Maintenance Tasks.

---

## Module 12: AWS Migration and Databases

- **Migration Strategies and Tools**
  - Lift-and-Shift vs. Re-Architecting.
  - AWS Migration Hub, Database Migration Service.
- **Working with Databases**
  - RDS (Relational Database Service): Setting Up and Managing Databases.
  - DynamoDB: NoSQL Database Basics.

---

Phase 3 : Advanced Concepts
Week 18

# Monitoring Tools

## Module 1: Introduction to Monitoring and Observability

- **Why Monitoring is Crucial?**
  - Importance of monitoring in DevOps.
  - Key differences: monitoring, logging, and observability.
- **Monitoring Tools Landscape**
  - Overview of Prometheus and Grafana.
  - Alternatives: Nagios, Zabbix, and Datadog.

---

## Module 2: Fundamentals of Prometheus

- **What is Prometheus?**
  - Origin, evolution, and core features.
  - Role in infrastructure monitoring.
- **Prometheus Architecture**
  - Components: Prometheus server, exporters, Alertmanager, Pushgateway.
  - Pull-based model and time-series database.

---

## Module 3: Prometheus Setup and Configuration

- **Installing Prometheus**
  - Installation on Docker and Kubernetes.
  - Configuring prometheus.yml for scraping metrics.
- **Data Collection**
  - Scraping metrics from default targets.
  - Adding and managing scrape jobs.

---

# Phase 3 : Advanced Concepts
# Week 18

# Monitoring Tools

## Module 4: Prometheus Query Language (PromQL)

- **Introduction to PromQL**
  - Querying and visualizing data.
  - Types of queries: instant, range, and aggregation.
- **Advanced PromQL Use Cases**
  - Functions: rate(), irate(), histogram_quantile().
  - Writing queries for custom metrics.

---

## Module 5: Exporters and Instrumentation

- **What are Exporters?**
  - Default exporters: Node Exporter, Blackbox Exporter.
  - Configuring exporters for system metrics.
- **Application Instrumentation**
  - Adding Prometheus client libraries to applications.
  - Exposing custom application metrics.

---

## Module 6: Alerting and Notifications with Prometheus

- **Prometheus Alertmanager**
  - Installing and configuring Alertmanager.
  - Writing alerting rules in Prometheus.
- **Notification Channels**
  - Integrating email, Slack, and PagerDuty for alerts.

---

# Phase 3 : Advanced Concepts
# Week 18

# Monitoring Tools

## Module 7: Introduction to Grafana

- **What is Grafana?**
  - Origin, key features, and use cases.
- **Grafana Architecture**
  - Data sources, plugins, and dashboards.
- **Connecting Grafana to Prometheus**
  - Adding Prometheus as a data source in Grafana.

---

## Module 8: Grafana Setup and Visualization

- **Installing Grafana**
  - Installation on Docker and Kubernetes.
  - Initial configuration and user management.
- **Creating Dashboards**
  - Setting up panels and queries.
  - Using templates and variables for dynamic dashboards.

---

## Module 9: Advanced Grafana Dashboards

- **Custom Visualization**
  - Using plugins for advanced charts.
  - JSON model for importing/exporting dashboards.
- **Grafana Alerts**
  - Creating and managing alerts on Grafana panels.
  - Configuring alert notification channels.

---

# Phase 3 : Advanced Concepts
# Week 18

# Monitoring Tools

## Module 10: Scaling and High Availability

- Scaling Prometheus
  - Federation and sharding techniques.
  - Best practices for high-availability setups.
- Scaling Grafana
  - Configuring Grafana for multiple users and teams.
  - Best practices for enterprise-scale dashboards.

---

## Module 11: Monitoring Kubernetes with Prometheus and Grafana

- Key Kubernetes Metrics
  - Metrics to monitor: pod health, resource usage, cluster status.
  - Tools: kube-state-metrics and cAdvisor.
- Deploying the Stack
  - Setting up Prometheus and Grafana in Kubernetes.
  - Visualizing Kubernetes metrics in Grafana.

---

## Module 12: Securing Monitoring Systems

- Authentication and Authorization
  - Setting up Grafana user roles and permissions.
  - Securing Prometheus endpoints.
- Best Practices
  - Using TLS/SSL for secure communication.
  - Managing sensitive data and credentials.

---

# DevOps Certification

## Complete with a certification


After completing the DevOps training, you can apply for our official “Certified DevOps Practitioner” digital Certificate. To qualify, you’ll need to complete the whole syllabus and submit the demo projects you worked on during the bootcamp.

This certificate serves as proof that you have mastered the skills taught in the DevOps Bootcamp and are capable of implementing end-to-end DevOps processes in a professional environment.
It is fully verifiable, making it an excellent addition to your LinkedIn profile and a valuable credential to share with future employers!