DevOps with AI

Transform your career with comprehensive DevOps training enhanced by cutting-edge AI agents. This intensive program covers everything from IT fundamentals to advanced cloud orchestration, equipping you with the skills demanded by today's tech industry. Master the complete DevOps lifecycle while leveraging artificial intelligence to accelerate your learning and productivity.

Fundamentals of IT & AI

Build a strong foundation in application lifecycle management, Agile methodologies, and AI concepts

Core & Advanced DevOps

Master CI/CD, containerization, Kubernetes, and infrastructure as code

Azure DevOps

Complete application lifecycle management with Microsoft's enterprise platform

AWS & Azure Cloud

Comprehensive cloud computing across both major platforms

Python for AI & DevOps

Programming fundamentals to advanced automation and data handling

GenAI & Agentic AI

Cutting-edge generative AI, LLMs, and autonomous agent systems

Digital Edify

India's First AI-Native Training Institute

Learn AI. Build Agents. Lead Future.

hello@digitaledify.ai

+91 6304982304

About Digital Edify

India's #1 Training Institute for the AI Era

Established: 2016 **Headquarters:** Hyderabad, Telangana **Reach:** Global (Online + Offline)

The Transformation Narrative

Digital Edify has evolved from a premium training institute in the Automation Era to an AI-first organisation leading the Agentic AI revolution. Since 2016, we've transformed over 100,000 professionals and built partnerships with more than 1,000 industry leaders. Our journey reflects the technological evolution of our time—from traditional job placement to career transformation, and now to building AI-native professionals who will shape the future of work.

1

Automation Era (2016-2023)

Premium Training Institute focused on job placement with 100K+ students trained

2

AI Revolution (2024-2025)

AI-Powered Training with industry-AI integration and career transformation focus

3

Agentic AI Leadership (2026+)

AI First Institute building AI-Native Professionals with 1 Million AI-Native Vision

> "We started in the Automation Era. We evolved through the AI Revolution. Now, we're leading the Agentic AI Future—with 100,000+ professionals already transformed and 1,000+ industry partners trusting our graduates."

Vision & Mission

Vision

"To Create 1 Million AI-Native Professionals Who Will Build the Agentic Future of Work"

Mission

"We transform learners into AI-native professionals through industry-aligned programmes that integrate Agentic AI into every discipline—from development to data science to enterprise platforms."

Course Highlights

Section 1: Fundamentals of IT & AI

Understand how modern applications, Agile practices, cloud computing, and AI fundamentals work together in real-world systems.

Section 2: Core DevOps & Advanced DevOps

Learn end-to-end DevOps practices including CI/CD, containers, Kubernetes, cloud infrastructure, monitoring, and AI-assisted automation.

Section 3: Azure DevOps (Application Lifecycle Management)

Manage Agile projects, source control, CI/CD pipelines, testing, and artifacts using Azure DevOps tools.

Section 4: Azure Cloud Computing

Build foundational cloud skills by deploying, managing, and governing applications on Microsoft Azure.

Section 5: AWS Cloud Computing

Gain hands-on experience with AWS compute, storage, networking, security, and cost optimization services.

Section 6: Python for AI & DevOps

Develop strong Python skills for automation, scripting, data handling, and DevOps workflows.

Section 7: Generative AI & Agentic AI

Design and deploy intelligent AI and agentic systems using LLMs, RAG pipelines, and production-ready AI workflows.

SECTION 1

Fundamentals of IT & AI

Module 1: Application Life Cycle Management

Understanding application development is the cornerstone of effective DevOps practice. This module introduces you to the complete spectrum of applications, from simple desktop programs to complex distributed web systems. You'll explore what defines an application, the various types that exist in modern computing environments, and the fundamental technologies that power them.

Agile & Scrum Framework

Module 2: Modern Project Management

The shift from traditional Waterfall to Agile methodologies represents one of the most significant transformations in software development history. While Waterfall follows a linear, sequential approach where each phase must be completed before the next begins, Agile embraces iterative development, continuous feedback, and adaptive planning. This module explores both methodologies in depth, helping you understand when each approach is appropriate and why Agile has become the dominant framework in modern DevOps environments.

Product Owner

Maximizes product value and manages the product backlog

Scrum Master

Facilitates the Scrum process and removes impediments

Development Team

Cross-functional professionals who deliver the increment

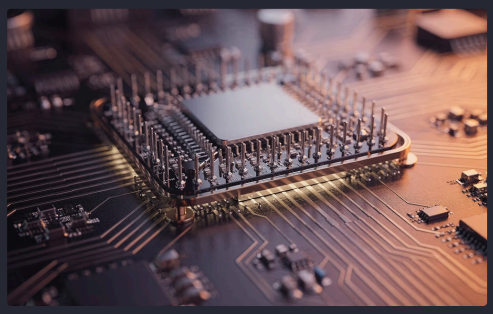
Scrum events provide the rhythm and structure for teams to collaborate effectively. You'll learn how Sprints create time-boxed iterations, how Sprint Planning sets clear goals, how Daily Scrums maintain alignment, how Sprint Reviews gather feedback, and how Sprint Retrospectives drive continuous improvement. Scrum artifacts—the Product Backlog, Sprint Backlog, and Increment—provide transparency and focus throughout the development process.

Writing effective user stories is an art that combines technical precision with business value. You'll master the format "As a [user], I want [goal], so that [benefit]," learning to create stories that are clear, testable, and valuable. Understanding Epics and Themes helps you organize large bodies of work, while Acceptance Criteria ensure everyone shares the same definition of "done." Estimation techniques, backlog management strategies, and practical tools like Google Sheets and Azure Boards round out your Scrum expertise, preparing you to lead or participate in high-performing Agile teams.

Computing & Data Fundamentals

Module 3: Infrastructure Essentials

Computing power forms the foundation of every digital service, application, and AI system in existence today. Understanding the technologies that provide this power is essential for any DevOps professional. This module begins with an exploration of why computing power matters—from enabling real-time data processing to supporting complex machine learning models that are transforming industries worldwide.



Central Processing Unit (CPU)

The brain of the computer, handling general-purpose computing tasks with sequential processing power. CPUs excel at complex logic and are essential for running operating systems and traditional applications.

Graphics Processing Unit (GPU)

Specialized processors designed for parallel computation, making them ideal for AI training, graphics rendering, and data-intensive workloads. GPUs can process thousands of operations simultaneously.

Infrastructure as a Service (IaaS)

Virtualized computing resources including servers, storage, and networking. You manage the operating system and applications while the provider handles the physical infrastructure.

Platform as a Service (PaaS)

Complete development and deployment environment in the cloud. Focus on building applications while the platform manages infrastructure, middleware, and runtime.

Software as a Service (SaaS)

Fully managed applications delivered over the internet. Users access software through web browsers without worrying about installation, maintenance, or infrastructure.

Introduction to AI, Generative AI & Agentic AI

Module 4: The AI Revolution

Artificial Intelligence represents one of the most transformative technologies of our era, fundamentally changing how we interact with computers and process information. At its core, AI enables machines to perform tasks that typically require human intelligence—learning from experience, recognizing patterns, making decisions, and solving complex problems. This module demystifies AI, explaining not just what it is, but how it actually works under the hood.

Machine Learning (ML) forms the foundation of modern AI systems. Rather than being explicitly programmed with rules, ML algorithms learn patterns from data, improving their performance over time. You'll explore supervised learning, where models learn from labeled examples; unsupervised learning, where algorithms discover hidden patterns; and reinforcement learning, where agents learn through trial and error. Understanding these fundamentals is crucial for anyone working in modern technology environments.

Deep Learning (DL) takes machine learning to the next level by using artificial neural networks inspired by the human brain. These multi-layered networks can automatically discover intricate patterns in vast amounts of data, enabling breakthroughs in image recognition, natural language processing, and speech synthesis. You'll understand how deep learning powers many of the AI applications you use daily, from voice assistants to recommendation systems.

Traditional AI

Rule-based systems and classical algorithms

Machine Learning

Pattern recognition from data

Deep Learning

Multi-layer neural networks

Generative AI

Creating new content and solutions

Real-World Applications

Module 5: Enterprise Systems in Action

Understanding how enterprise applications work in the real world bridges the gap between theoretical knowledge and practical DevOps skills. This module explores four critical categories of business applications that DevOps professionals frequently encounter, deploy, and maintain. Each system type presents unique challenges and requirements that will shape your approach to infrastructure, deployment strategies, and operational excellence.

Customer Relationship Management (CRM)

CRM systems like Salesforce, HubSpot, and Microsoft Dynamics serve as the central nervous system for customer interactions.

Human Resource Management Systems (HRMS)

HRMS platforms like Workday, SAP SuccessFactors, and BambooHR manage the complete employee lifecycle—from recruitment and onboarding to payroll, benefits, performance reviews, and offboarding.

Retail & E-Commerce Applications

E-commerce platforms like Shopify, Magento, and custom-built solutions power the global digital economy. These applications must handle traffic spikes during sales events, process payments securely, manage inventory in real-time, and provide seamless user experiences across devices.

Healthcare Applications

Healthcare systems like Epic, Cerner, and telemedicine platforms manage patient records, appointment scheduling, prescription management, and clinical workflows. These applications operate under strict regulatory frameworks including HIPAA, requiring exceptional security, audit trails, and disaster recovery capabilities. .

SECTION 2

Core DevOps & Advanced DevOps

Module 1: DevOps Fundamentals & Server Essentials

DevOps represents a cultural and technical revolution in how software is built, deployed, and maintained. This module introduces you to the core philosophy that breaks down traditional silos between development and operations teams, fostering collaboration, automation, and continuous improvement. You'll understand the DevOps infinity loop—a continuous cycle of planning, coding, building, testing, releasing, deploying, operating, and monitoring—and how it integrates seamlessly with the Software Development Life Cycle.

Server fundamentals form the backbone of any infrastructure. You'll explore three critical server types: Web servers that handle HTTP requests and serve content to users, Database servers that store and manage data with ACID compliance, and Application servers that execute business logic and process transactions. Understanding how these servers work together in a typical three-tier architecture is essential for designing robust, scalable systems.

Cloud Computing Introduction

Explore AWS and Azure platforms, understanding their core services, pricing models, and global infrastructure that enables worldwide deployment

Networking Basics

Master VPCs, subnets, security groups, ports, and protocols—the fundamental building blocks of cloud networking and security

Hands-On Practice

Launch and configure EC2 instances on AWS and Virtual Machines on Azure, gaining practical experience with cloud infrastructure

Networking knowledge is non-negotiable for DevOps professionals. You'll learn about Virtual Private Clouds (VPCs) that provide isolated network environments, Security Groups that act as virtual firewalls, and the ports and protocols that enable communication between services. This foundational knowledge ensures you can design secure, efficient network architectures that protect resources while enabling necessary connectivity.

Linux Essentials & Shell Scripting

Module 2: The DevOps Operating System

Linux dominates the server landscape, powering the vast majority of web servers, cloud instances, and containerized applications worldwide. Understanding why Linux is essential for DevOps goes beyond its prevalence—it's about the philosophy of open-source collaboration, the power of command-line efficiency, and the flexibility that makes Linux the platform of choice for modern infrastructure.

01

File & Directory Management

Create, move, copy, and delete files and directories with precision

02

User & Group Management

Add users, create groups, and manage access control

03

Permissions & Ownership

Master chmod and chown for security and access control

04

Package Management

Install, update, and remove software using APT

05

Shell Scripting

Automate repetitive tasks with Bash scripts

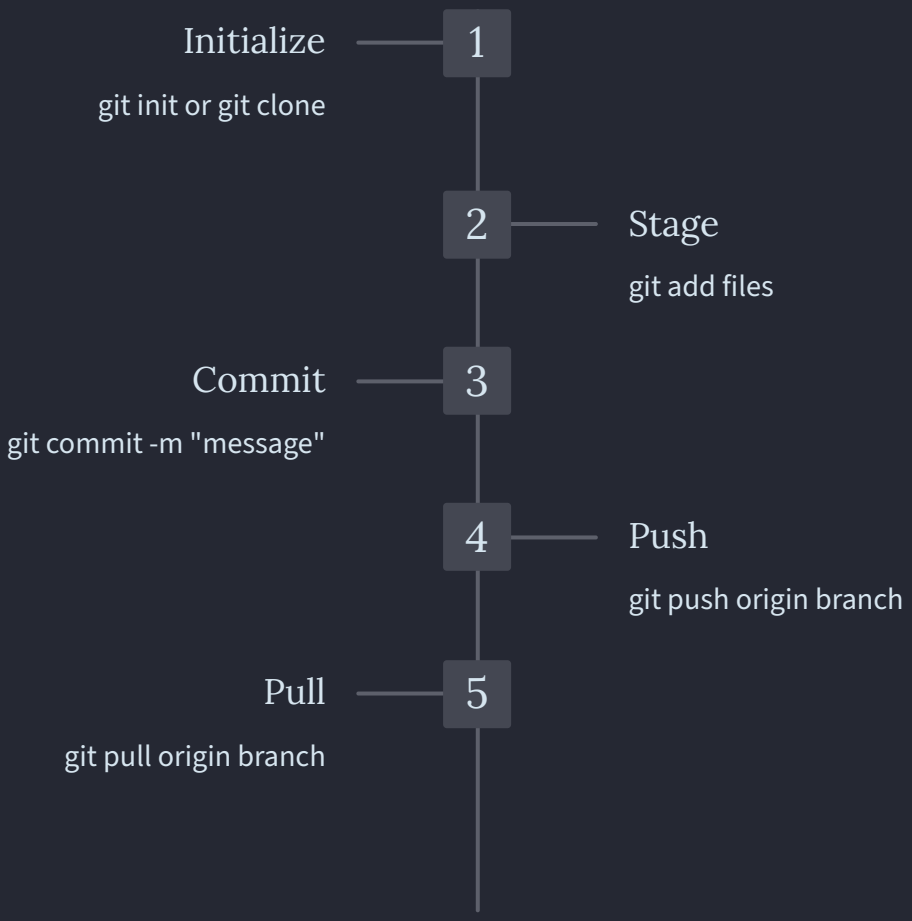
Essential CLI commands become second nature through practice. You'll learn file manipulation with cat, grep, sed, and awk; process management with ps, top, and kill; and system monitoring with df, du, and free. User and group management ensures proper access control, while file permissions and ownership using chmod and chown protect sensitive data and maintain system security.

Key Technologies: Ubuntu Linux, Bash, vi/vim text editor, APT package manager, Shell Scripting for automation

Version Control with Git & GitHub

Module 3: Collaborative Development

Version control is the time machine of software development, allowing teams to track every change, collaborate without conflicts, and revert mistakes instantly. Git has become the de facto standard for version control, and GitHub serves as the social network for code, hosting millions of repositories and facilitating global collaboration. This 10-12 hour module with 8 hands-on labs transforms you from a Git novice to a confident practitioner.



Basic Git operations form your daily workflow. You'll master git add to stage changes, git commit to save snapshots with descriptive messages, git push to share your work with remote repositories, and git pull to incorporate others' changes. These commands become muscle memory through repeated practice in the lab exercises.

Pull Requests	Code Reviews
Webhooks	Forking Workflow

Application Development Fundamentals

Module 4: Building Real-World Applications

The Software Development Life Cycle (SDLC) provides the roadmap for building quality software. You'll apply SDLC principles throughout the module, from planning your LMS features to deploying the final application. Three-tier architecture separates concerns into presentation (frontend), business logic (backend), and data storage (database) layers. This separation enables independent scaling, easier maintenance, and clearer team responsibilities.

Database Layer

Set up PostgreSQL, design schemas, create tables, and establish relationships for your LMS data model

Backend Layer

Build RESTful APIs with Node.js and Express, implement authentication, and connect to the database

Frontend Layer

Create responsive user interfaces with ReactJS, manage state, and consume backend APIs

Web Server

Configure NGINX as a reverse proxy, enable SSL, and optimize performance

Process Management

Use PM2 to keep your Node.js application running, enable auto-restart, and monitor performance

Key Technologies: PostgreSQL database, Node.js runtime, Express framework, ReactJS library, NGINX web server, PM2 process manager, NPM package manager

Continuous Integration & Continuous Deployment

Module 5: Automating the Software Pipeline

CI/CD represents the automation backbone of modern DevOps, transforming how software moves from developer laptops to production servers. Continuous Integration ensures code changes are automatically built and tested, catching bugs early when they're cheapest to fix. Continuous Deployment takes this further, automatically releasing validated changes to production, enabling teams to deliver value to users faster than ever before.

Source Control Code pushed to GitHub triggers the pipeline	**Build** Jenkins compiles code and resolves dependencies
Test Automated tests verify functionality and quality	**Analyze** SonarQube scans for code quality issues
Artifact Nexus stores versioned build artifacts	**Deploy** Application released to target environment

SonarQube Integration

Static code analysis with SonarQube identifies bugs, vulnerabilities, and code smells before they reach production. You'll configure quality gates that automatically fail builds when code doesn't meet standards, enforcing quality across your team.

Nexus Repository

Artifact management with Nexus Repository provides a central location for storing build outputs, dependencies, and Docker images. You'll implement versioning strategies and retention policies that balance storage costs with traceability needs.

The module culminates in building a complete CI/CD pipeline for your LMS application. Every code push triggers automated builds, runs comprehensive tests, performs code quality analysis, stores artifacts in Nexus, and deploys to your target environment. This hands-on experience demonstrates the power of automation, showing how CI/CD reduces manual errors, accelerates delivery, and increases confidence in releases.

Containerization with Docker

Module 6: Packaging Applications for Portability

Understanding the difference between containers and virtual machines is fundamental. Virtual machines include a full operating system, making them heavy and slow to start. Containers share the host OS kernel, making them lightweight, fast, and efficient. A single server can run dozens of containers but only a handful of VMs, dramatically improving resource utilization and reducing costs.

Monolithic

Single large application, tightly coupled components, difficult to scale

Microservices

Small independent services, loosely coupled, easy to scale and maintain

Containerized

Each microservice in its own container, portable and consistent

Monolithic versus Microservices architecture represents a fundamental shift in application design. Monoliths bundle all functionality into a single application, making them simple to develop initially but difficult to scale and maintain. Microservices decompose applications into small, independent services that communicate via APIs. Containers are the perfect packaging for microservices, providing isolation while maintaining efficiency.

1

Bridge Network

Default network for containers on a single host, enabling inter-container communication

2

Host Network

Container shares the host's network stack, offering maximum performance

3

Overlay Network

Connects containers across multiple Docker hosts, essential for distributed applications

Key Technologies: Docker Engine, Dockerfile, Docker Compose, Docker Hub, Docker Volumes, Multi-stage builds

Container Orchestration with Kubernetes

Module 7: Managing Containers at Scale

Master Node Components

The API Server handles all requests, the Scheduler assigns Pods to Nodes, the Controller Manager maintains desired state, and etcd stores cluster configuration. These components work together to orchestrate your entire cluster.

Worker Node Components

The Kubelet manages Pods on each node, the Container Runtime (Docker/containerd) runs containers, and kube-proxy handles networking. Worker nodes execute your actual workloads, running the containers that serve your applications.

01

ConfigMaps

Store non-sensitive configuration data as key-value pairs

02

Secrets

Securely store sensitive information like passwords and API keys

03

Persistent Volumes

Provide durable storage that survives Pod restarts

04

Ingress Controllers

Route external HTTP/HTTPS traffic to internal services

05

Namespaces

Organize resources and provide isolation between teams

ConfigMaps and Secrets management separates configuration from code, enabling the same container image to run in different environments. Persistent storage in Kubernetes abstracts underlying storage systems, providing a consistent interface whether you're using local disks, network storage, or cloud volumes. Service discovery and load balancing happen automatically—Kubernetes maintains DNS records and distributes traffic intelligently.

Kubernetes Production with AWS EKS

Module 8: Enterprise-Grade Container Orchestration

AWS EKS cluster architecture separates the control plane (managed by AWS) from the data plane (your worker nodes).

Cluster Creation

Use eksctl or AWS Console to provision your EKS cluster with proper networking and IAM configuration

Node Groups

Add managed or self-managed node groups, choosing instance types and scaling parameters

Load Balancer

Install AWS Load Balancer Controller to automatically provision ALBs and NLBs for services

Application Deploy

Deploy your containerized applications with proper resource requests and limits

Monitor & Optimize

Use CloudWatch and Kubernetes metrics to monitor performance and optimize cos

EKS Networking

EKS networking uses AWS VPC CNI, giving each Pod a real VPC IP address. This enables direct communication with other AWS services and simplifies security group rules.

IAM for Service Accounts

IAM Roles for Service Accounts (IRSA) provides fine-grained permissions to Pods without sharing credentials.

Infrastructure as Code with Terraform

Module 9: Automating Infrastructure Provisioning

Infrastructure as Code (IaC) treats infrastructure configuration as software, bringing version control, code review, and automation to infrastructure management. Terraform has become the leading IaC tool, supporting multiple cloud providers with a consistent workflow. This module teaches you to define, provision, and manage infrastructure using declarative configuration files, eliminating manual console clicking and ensuring reproducible deployments.

Providers

Plugins that enable Terraform to interact with cloud platforms (AWS, Azure, GCP), SaaS services, and other APIs. Each provider offers resources and data sources specific to that platform.

Resources

Infrastructure components you want to create—EC2 instances, S3 buckets, VPCs, databases. Resources are the building blocks of your infrastructure, defined with type and configuration.

Variables

Input parameters that make configurations reusable across environments. Variables enable the same code to deploy development, staging, and production with different values.

Outputs

Values extracted from your infrastructure after creation—IP addresses, DNS names, resource IDs. Outputs enable automation and integration with other tools.

HCL (HashiCorp Configuration Language) provides a human-readable syntax for defining infrastructure. You'll write resource blocks that declare desired infrastructure, use variables for flexibility, and leverage data sources to query existing infrastructure. Terraform providers support AWS, Azure, GCP, and hundreds of other services, enabling multi-cloud deployments with a single tool and consistent workflow.

SECTION 3

Azure DevOps

Module 1: Introduction to Azure DevOps

Azure DevOps provides a comprehensive suite of tools for managing the complete application lifecycle, from planning and development to testing and deployment. Unlike point solutions that require integration, Azure DevOps offers a unified platform where all tools work seamlessly together. This module introduces the five core services that make Azure DevOps a powerful choice for teams of any size.

Azure Boards

Agile project management with work items, sprints, backlogs, and customizable workflows. Track work from epic to task with full traceability.

Azure Repos

Unlimited private Git repositories with pull requests, code reviews, and branch policies. Enterprise-grade version control with fine-grained permissions.

Azure Pipelines

CI/CD automation supporting any language, platform, and cloud. Build, test, and deploy with YAML pipelines or classic visual designer.

Azure Test Plans

Manual and exploratory testing with test case management, execution tracking, and integration with work items and pipelines.

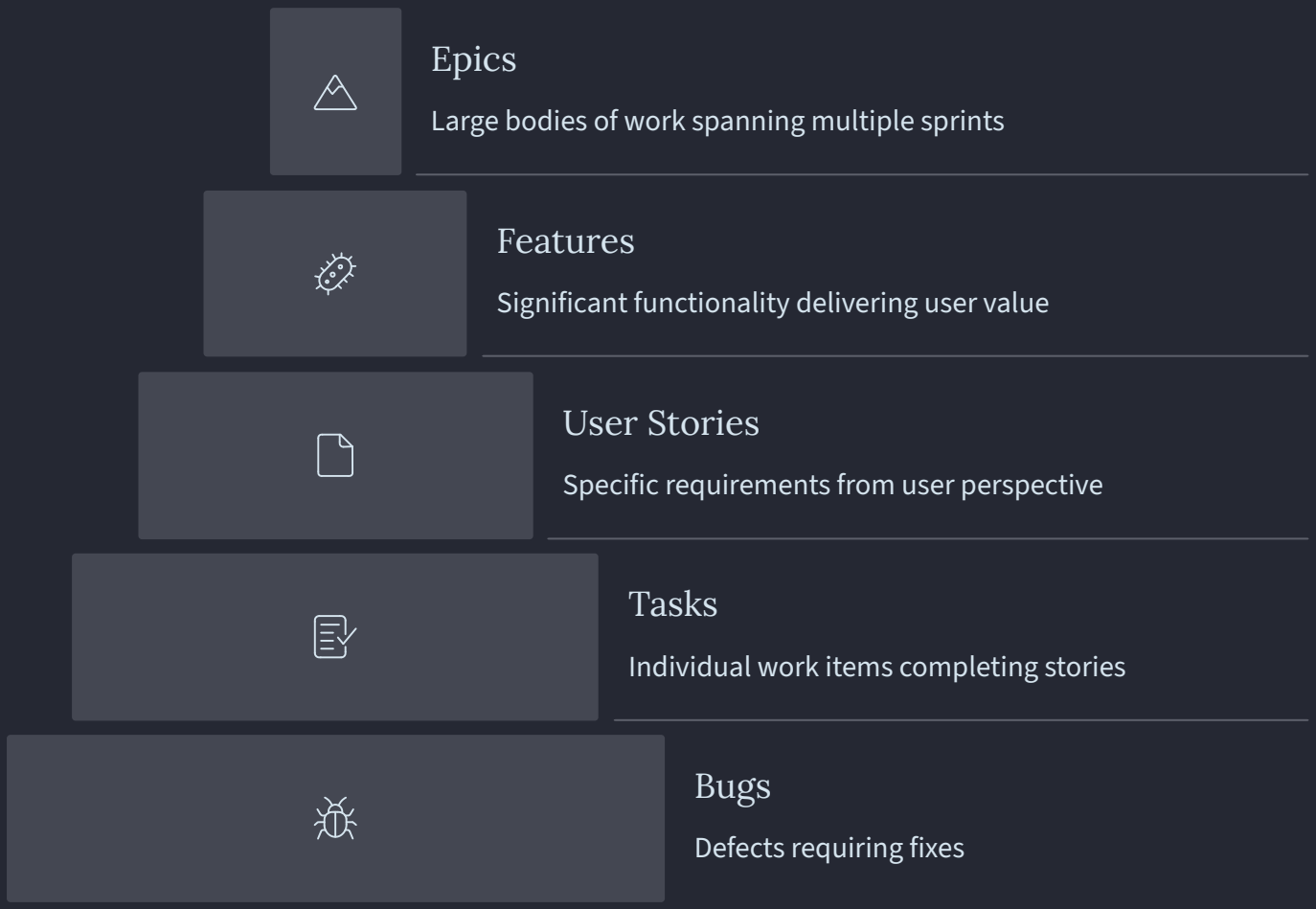
Azure Artifacts

Package management for Maven, npm, NuGet, and Python. Host private feeds and share packages across teams with upstream sources.

Azure DevOps integrates deeply with Microsoft's ecosystem—Active Directory for authentication, Visual Studio for development, and Azure cloud for deployment. However, it's not limited to Microsoft technologies. Azure DevOps supports Linux, macOS, and Windows; works with any programming language; and deploys to AWS, GCP, or on-premises infrastructure just as easily as Azure.

Agile Project Management with Azure Boards

Module 2: Planning and Tracking Work



Work Items and Sprints

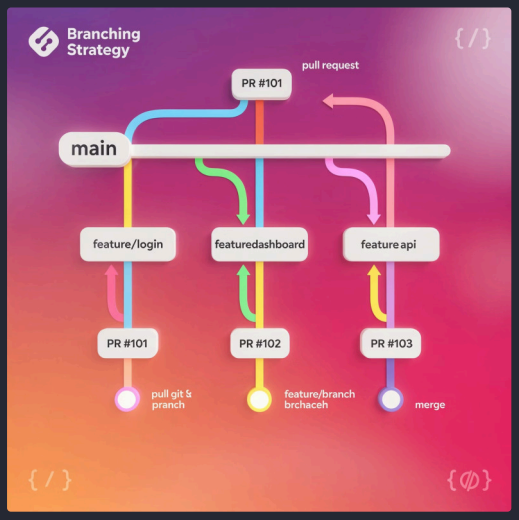
Managing tasks and sprints in Azure Boards follows Scrum principles. You'll plan sprints by moving work items from the backlog, assign tasks to team members, track progress with burndown charts, and conduct retrospectives to improve processes. Sprint planning becomes a collaborative exercise where the team commits to achievable goals.

Azure Boards Integrations

Enhance Azure Boards with integrations for Slack (notifications), Microsoft Teams (collaboration), GitHub (code linking), and custom webhooks (automation). These integrations keep everyone informed without requiring constant board checking, improving team communication and reducing context switching.

Version Control with Azure Repos

Module 3: Enterprise Git Management



Azure Repos provides enterprise-grade Git repository hosting with unlimited private repositories, advanced security features, and seamless integration with Azure Pipelines. While Git fundamentals remain the same, Azure Repos adds enterprise capabilities like branch policies, required

SECTION 4

Azure Cloud Computing

Module 1: Introduction to Cloud Concepts and Azure

Cloud computing has fundamentally transformed how organizations build and operate IT infrastructure. Rather than purchasing and maintaining physical servers, companies now rent computing resources on-demand, paying only for what they use. This module introduces core cloud concepts and Microsoft Azure, one of the three major cloud platforms alongside AWS and Google Cloud. Understanding these fundamentals is essential for any modern IT professional.

Infrastructure-as-a-Service (IaaS)

Rent virtual machines, storage, and networks. You manage the operating system, middleware, and applications. Maximum control and flexibility. Examples: Azure Virtual Machines, Azure Storage.

Platform-as-a-Service (PaaS)

Focus on application development while the platform manages infrastructure, OS, and runtime. Faster development with less operational overhead. Examples: Azure App Service, Azure SQL Database.

Software-as-a-Service (SaaS)

Use fully managed applications over the internet. No infrastructure or application management required. Examples: Microsoft 365, Dynamics 365, Salesforce.

Differentiating between Public Cloud, Private Cloud, and Hybrid Cloud models helps you choose the right approach. Public Cloud (Azure, AWS, GCP) offers resources to anyone over the internet with maximum scalability and minimum management. Private Cloud runs on dedicated infrastructure, providing maximum control and security for sensitive workloads. Hybrid Cloud combines both, enabling workloads to move between public and private clouds based on requirements, costs, and compliance needs. Many organizations adopt hybrid approaches, keeping sensitive data on-premises while leveraging public cloud for scalability and innovation.

Core Azure Services

Module 2: Compute, Storage, Networking, and Databases

Azure Compute

Azure compute services provide processing power for your applications. Virtual Machines offer full control over the operating system and configuration. App Service provides managed hosting for web apps and APIs without managing infrastructure. Azure Functions enable serverless computing where you pay only for execution time. Container Instances and Azure Kubernetes Service run containerized applications. Choose based on control needs, scaling requirements, and operational preferences.

Azure Storage

Azure storage services handle different data types with optimized solutions. Blob Storage stores unstructured data like images, videos, and backups. File Storage provides SMB file shares accessible from cloud or on-premises. Queue Storage enables reliable messaging between application components. Table Storage offers NoSQL key-value storage. Disk Storage provides persistent disks for virtual machines. Each service is optimized for specific scenarios with different performance and cost characteristics.

Azure SQL Database

Fully managed relational database based on SQL Server. Automatic backups, patching, and high availability. Ideal for transactional applications requiring ACID compliance.

Cosmos DB

Globally distributed NoSQL database with multiple APIs (SQL, MongoDB, Cassandra). Millisecond latency and automatic scaling. Perfect for globally distributed applications.

Azure Database for MySQL/PostgreSQL

Managed open-source databases with built-in high availability, automated backups, and security. Ideal for applications already using these databases.

Azure Pricing, Support, and Governance

Module 3: Managing Costs and Compliance

Azure pricing varies by service, region, and usage patterns. Virtual machines charge by the hour based on size and operating system. Storage charges by capacity and transactions. Bandwidth charges for data egress (outbound transfer). Many services offer free tiers for development and testing. Reserved Instances provide significant discounts (up to 72%) for one or three-year commitments. Spot Instances offer even deeper discounts for interruptible workloads.

72%

Reserved Instance Savings

Commit to one or three years for maximum discounts

90%

Spot Instance Savings

Use spare capacity for fault-tolerant workloads

5

Support Plans

From free basic support to enterprise-level assistance

Role-Based Access Control (RBAC)

Grant users only the permissions they need using built-in or custom roles. RBAC applies at subscription, resource group, or individual resource levels, enabling fine-grained access control that follows the principle of least privilege.

Resource Locks

Prevent accidental deletion or modification of critical resources. CanNotDelete locks prevent deletion while allowing modifications. ReadOnly locks prevent both deletion and modifications. Locks override RBAC permissions, providing an additional safety layer.

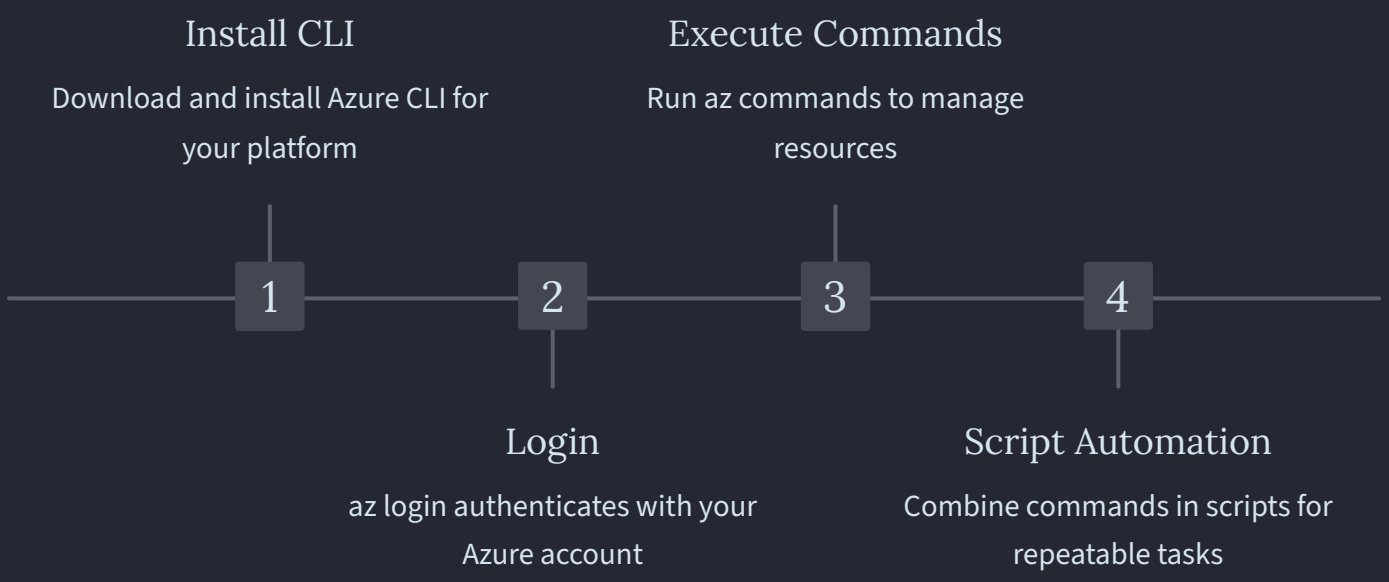
Azure Policy

Enforce organizational standards and compliance requirements automatically. Policies can require specific tags, restrict resource locations, enforce naming conventions, or mandate security configurations. Policies apply to new and existing resources, ensuring consistent governance.

Managing Azure Resources

Module 4: Portal, CLI, and Management Tools

Azure provides multiple interfaces for managing resources, each suited to different tasks and preferences. The Azure Portal offers a graphical interface ideal for exploration and one-time tasks. Azure CLI provides command-line efficiency for automation and scripting. Azure PowerShell serves Windows administrators familiar with PowerShell syntax. This module teaches you to use these tools effectively, choosing the right interface for each situation.



Azure Command-Line Interface (CLI) provides cross-platform command-line access to Azure services. Available on Windows, macOS, and Linux, Azure CLI uses intuitive command syntax: az [service] [operation] [parameters]. For example, az vm create provisions a virtual machine, az storage account list shows storage accounts, and az group delete removes a resource group. CLI commands can be combined in shell scripts for automation, making them ideal for DevOps workflows and CI/CD pipelines.

Azure Monitor

Collects and analyzes telemetry from Azure resources and applications. Monitor provides metrics (numerical time-series data), logs (text-based event data), and alerts (notifications when conditions are met). Application Insights monitors application performance, while Log Analytics queries log data for troubleshooting and analysis.

Azure Resource Manager

The deployment and management service for Azure, providing a consistent management layer. ARM templates define infrastructure as code using JSON, enabling repeatable deployments. Resource Manager handles authentication, authorization, and resource lifecycle management, ensuring consistent operations across all Azure tools.

Azure Application Services

Module 5: App Services and Advanced Topics

Azure App Service provides a fully managed platform for building, deploying, and scaling web applications and APIs. Unlike Infrastructure-as-a-Service where you manage virtual machines, App Service abstracts infrastructure concerns, allowing you to focus entirely on application code. This module explores App Service capabilities, from basic web hosting to advanced features like deployment slots, auto-scaling, and container support.

Virtual Network Integration

Connect App Service to your virtual network, enabling access to resources in VNets or on-premises networks via VPN or ExpressRoute. This integration maintains security while accessing databases, APIs, or other services not exposed to the internet.

Private Endpoints

Expose your App Service only within your virtual network using private endpoints. Traffic never traverses the public internet, meeting strict security and compliance requirements. Ideal for internal applications or APIs consumed by other Azure services.

Hybrid Connections

Access on-premises resources without VPN by creating secure tunnels to specific TCP endpoints. Hybrid Connections enable gradual cloud migration, allowing cloud applications to access on-premises databases or services during transition periods.

Networking for App Service provides multiple options for controlling traffic flow and accessing resources. Understanding these networking features enables designing secure, compliant architectures that meet enterprise requirements.

SECTION 5

AWS Cloud Computing

Module 1: AWS Foundations

Amazon Web Services (AWS) pioneered cloud computing and remains the market leader with the most comprehensive service portfolio. This foundational module introduces AWS core concepts, global infrastructure, and essential services that form the basis of cloud solutions. Understanding AWS fundamentals is critical for any cloud professional, as AWS patterns and practices influence the entire industry.

Regions

Geographic areas containing multiple availability zones

Availability Zones

Isolated data centers within regions for high availability

Edge Locations

Content delivery network endpoints for low-latency access

01

Account Setup

Create AWS account and configure billing alerts

02

IAM Configuration

Set up users, groups, roles, and policies for security

03

VPC Creation

Design and deploy virtual private cloud networking

04

CLI Setup

Install and configure AWS CLI for automation

05

Monitoring

Configure CloudWatch and CloudTrail for visibility

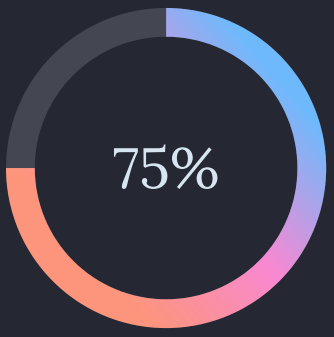
AWS Compute Services

Module 2: EC2, Auto Scaling, Lambda, and Load Balancing

AWS compute services provide the processing power for your applications, from traditional virtual machines to serverless functions. This module covers Amazon EC2 (Elastic Compute Cloud) for flexible virtual servers, Auto Scaling for dynamic capacity management, Elastic Load Balancing for distributing traffic, AWS Lambda for serverless computing, and API Gateway for building APIs. Mastering these services enables building scalable, resilient applications that handle varying workloads efficiently.

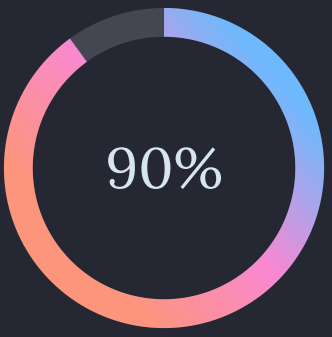
Instance Types

Optimized for different workload requirements



Reserved Savings

Discount for one or three-year commitments



Spot Savings

Discount for interruptible workloads

Traffic Arrives

Users send requests to your application

Load Balancer

Distributes traffic across healthy instances

EC2 Instances

Process requests and return responses

Auto Scale

Add or remove instances based on load

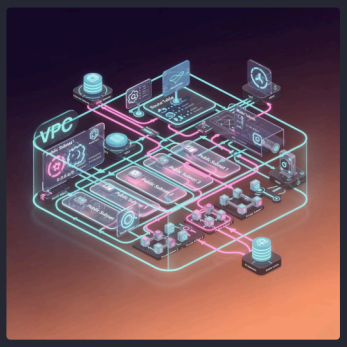
hello@digitaledify.ai

+91 6304982304

Digital Edify

AWS Networking and Security

Module 3: VPC, KMS, Cognito, and Security Best Practices



AWS networking and security services provide the foundation for building secure, isolated cloud environments. Amazon VPC (Virtual Private Cloud) creates logically isolated networks, AWS KMS (Key Management Service) manages encryption keys, and Amazon Cognito handles user authentication. This module teaches you to design secure network architectures and implement defense-in-depth security strategies.

Network ACLs

Stateless firewalls at the subnet level, controlling inbound and outbound traffic with allow and deny rules

Security Groups

Stateful firewalls at the instance level, allowing only specified traffic and denying everything else by default

VPC Peering

Private connections between VPCs, enabling resources to communicate as if in the same network

VPN Connections

Encrypted tunnels between your VPC and on-premises networks for hybrid cloud architectures

Identity Security

Use IAM roles instead of access keys, enable MFA for all users, follow least privilege principle, and rotate credentials regularly

Resource Security

Encrypt data at rest and in transit, use security groups restrictively, enable VPC Flow Logs, and implement defense in depth

Monitoring & Compliance

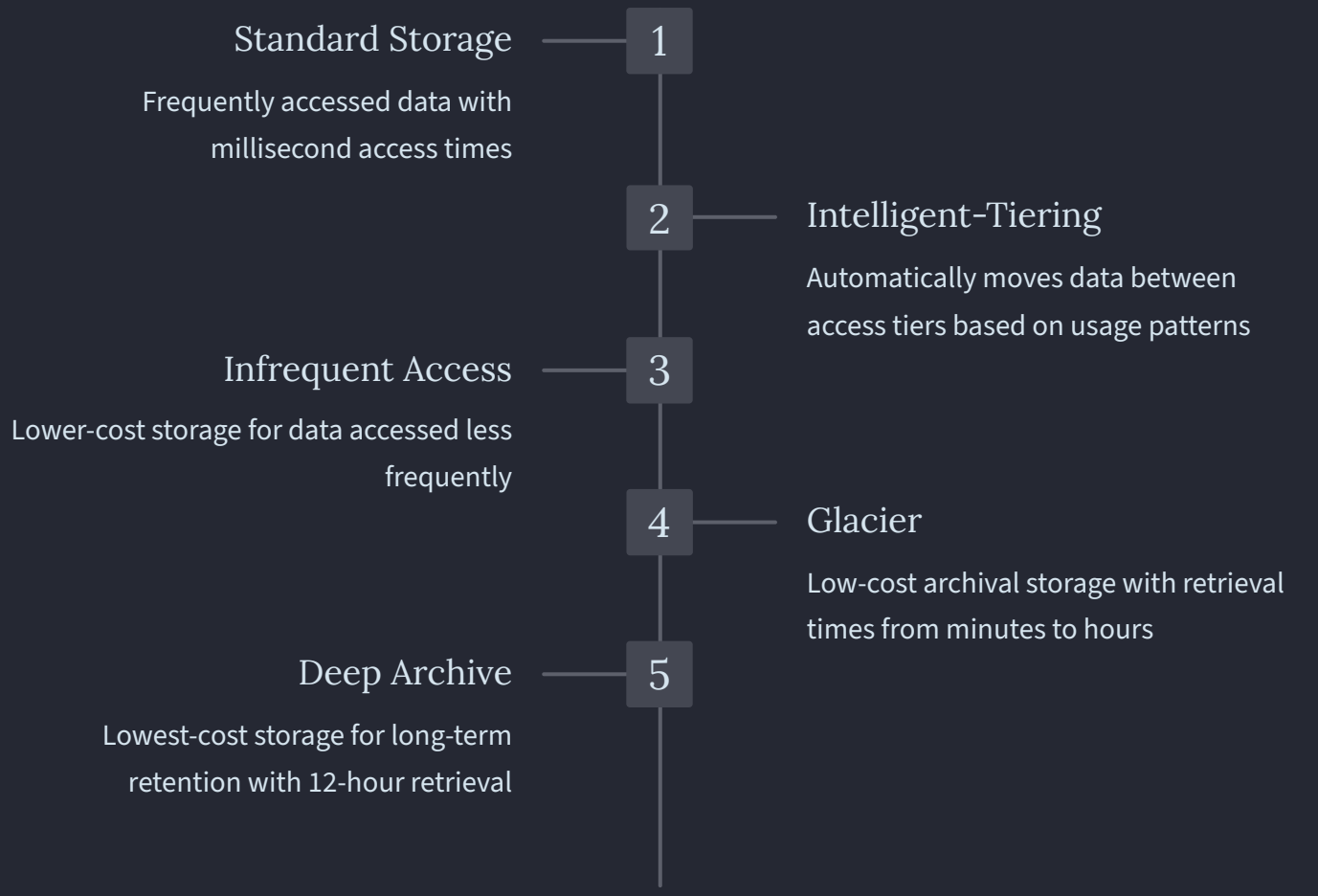
Enable CloudTrail for all regions, use AWS Config for compliance, implement AWS GuardDuty for threat detection, and conduct regular security audits

AWS Storage and Cost Management

Module 4 & 5: S3, EBS, Cost Optimization

AWS storage services provide durable, scalable solutions for every data type and access pattern. Amazon S3 offers object storage for unstructured data, while Amazon EBS provides block storage for EC2 instances. This module covers storage fundamentals and cost management strategies, teaching you to choose appropriate storage services and optimize spending across your AWS environment.

Amazon S3 Basics introduce object storage where data is stored as objects within buckets. Each object consists of data, metadata, and a unique key. Buckets are globally unique containers that organize objects. S3 provides 99.999999999% (11 nines) durability by automatically replicating data across multiple facilities. Permissions control access at bucket and object levels using bucket policies, IAM policies, and Access Control Lists (ACLs). S3 supports hosting static websites, serving as a data lake for analytics, and backing up critical data.



Amazon Elastic Block Store (EBS) provides persistent block storage for EC2 instances, functioning like virtual hard drives. EBS volumes attach to EC2 instances and persist independently of instance lifecycle. Volume types include General Purpose SSD (balanced price and performance), Provisioned IOPS SSD (high-performance for databases), Throughput Optimized HDD (low-cost for big data), and Cold HDD (lowest cost for infrequent access). Snapshots create point-in-time backups stored in S3, enabling disaster recovery and volume cloning. Creating and managing volumes involves choosing the appropriate type, size, and IOPS based on workload requirements.

AWS Cost Explorer

Visualize and analyze spending patterns over time. Filter by service, region, or tag to identify cost drivers. Forecast future costs based on historical trends. Create custom reports for different stakeholders. Cost Explorer helps you understand where money is going and identify optimization opportunities.

AWS Budgets

Set custom cost and usage budgets with alerts when thresholds are exceeded. Create budgets for specific services, accounts, or tags. Receive notifications via email or SNS when spending approaches or exceeds limits. Budgets prevent surprise bills and enable proactive cost management.

AWS Trusted Advisor provides real-time guidance to help optimize your AWS environment across cost optimization, performance, security, fault tolerance, and service limits. Trusted Advisor checks include identifying idle resources, recommending reserved instances, detecting security vulnerabilities, and suggesting performance improvements. AWS Savings Plans offer flexible pricing across EC2, Lambda, and Fargate, providing up to 72% savings compared to on-demand pricing. Savings Plans require commitment to consistent usage (measured in dollars per hour) for one or three years, offering significant discounts while maintaining flexibility to change instance types, regions, or operating systems. Cost optimization combines right-sizing resources, using appropriate pricing models, implementing lifecycle policies, and continuously monitoring spending to maximize AWS value.

SECTION 6

Python for AI & DevOps

Module 1: Python Fundamentals

Python has become the lingua franca of DevOps automation and AI development, valued for its readability, extensive libraries, and versatility. This comprehensive module introduces Python from the ground up, covering everything from basic syntax to control flow structures. Whether you're automating infrastructure tasks or building AI applications, Python provides the foundation for modern technical work.

01

Syntax & Keywords

Learn Python's 35 reserved keywords and basic syntax rules

02

Variables & Types

Understand dynamic typing, memory management, and data types

03

Operators

Master arithmetic, comparison, logical, and assignment operators

04

Control Flow

Use conditionals and loops to control program execution

05

User Input

Interact with users through the input() function

Operators provide the building blocks for expressions. Arithmetic operators (+, -, *, /, //, %, **) perform mathematical operations. Comparison operators (==, !=, <, >, <=, >=) compare values. Logical operators (and, or, not) combine boolean expressions. Assignment operators (=, +=, -=, *=, /=) assign and modify variables. Conditional Statements (if, elif, else, match-case) enable decision-making. Loops (while, for) repeat code blocks, with range() generating number sequences. Control Flow statements (break, continue, pass) modify loop behavior. User Input with input() function enables interactive programs, reading text from users and returning strings that can be converted to other types as needed.

String Manipulation

Module 2: Working with Text Data

Strings are fundamental to programming, representing text data from user input to log files to API responses. Python provides rich string manipulation capabilities that make text processing elegant and efficient. This module covers everything from basic string operations to advanced formatting techniques, essential skills for DevOps automation and data processing.

String Definition and Rules in Python use single quotes ('text'), double quotes ("text"), or triple quotes ('''text''' or """text""") for multi-line strings. Strings are immutable—once created, they cannot be changed, though you can create new strings based on existing ones. This immutability ensures string operations are safe and predictable.

Indexing

Access individual characters using positive indices (0, 1, 2...) from the start or negative indices (-1, -2, -3...) from the end

Slicing

Extract substrings using [start:end:step] syntax. Omit start for beginning, end for end, or step for default increment of 1

Operations

Concatenate strings with +, repeat with *, check membership with in, and get length with len()

String Formatting provides multiple approaches for creating formatted strings. F-strings (f"Hello {name}") offer the most readable syntax, embedding expressions directly in strings. The format() method provides more control with positional and keyword arguments. Old-style % formatting remains common in legacy code. Choose f-strings for new code—they're faster, more readable, and support complex expressions.

Case Conversion

upper(), lower(), capitalize(), title(), swapcase() transform string case for normalization or display

Search Methods

find(), index(), count() locate substrings. find() returns -1 if not found, index() raises exception

Checking Methods

isalpha(), isdigit(), isalnum(), isspace() validate string content for input validation

Trimming Methods

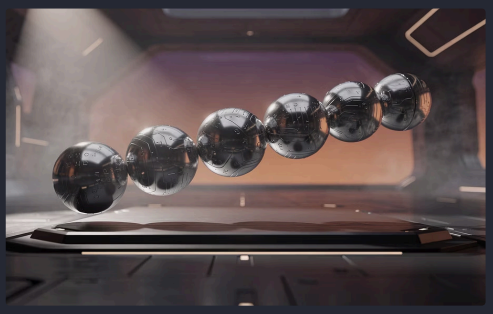
strip(), lstrip(), rstrip() remove whitespace from ends, crucial for cleaning user input

Data Structures - Lists & Tuples

Module 3: Ordered Collections

Lists and tuples are Python's fundamental ordered collections, storing sequences of items that can be accessed by position. Lists are mutable (changeable), while tuples are immutable (unchangeable). This module teaches you to work with both data structures effectively, understanding when to use each and mastering the methods that make Python collections so powerful.

Simple versus Complex Data Types distinguishes between single values (integers, floats, strings) and collections that hold multiple values (lists, tuples, dictionaries, sets). Collections enable organizing related data, iterating over multiple items, and building sophisticated data structures. Understanding collections is essential for processing data at scale.



Lists: Creation, Indexing, Slicing

Create lists using square brackets: [1, 2, 3] or list(). Access elements by index (positive or negative). Slice lists to extract sublists using [start:end:step]. Lists can contain mixed types and nested lists, enabling complex data structures. Indexing and slicing work identically to strings, making Python consistent and learnable.

Tuples: Creation and Operations

Create tuples using parentheses: (1, 2, 3) or tuple(). Single-element tuples require trailing comma: (1,). Tuples support indexing and slicing like lists but cannot be modified after creation. Use tuples for data that shouldn't change—coordinates, RGB colors, function return values. Tuple immutability enables using them as dictionary keys.

Tuple Packing

Create tuples without parentheses: coordinates = 10, 20, 30. Python automatically packs values into a tuple.

Tuple Unpacking

Assign tuple elements to variables: x, y, z = coordinates. Number of variables must match tuple length.

Lists vs Tuples

Lists: mutable, slower, more memory. Tuples: immutable, faster, less memory. Use lists for changing data, tuples for fixed data.

Data Structures - Dictionaries & Sets

Module 4: Key-Value and Unique Collections

Dictionaries: Creation and Access use curly braces with key-value pairs: {'name': 'Alice', 'age': 30}. Access values using keys: person['name']. Dictionaries provide O(1) average-case lookup time, making them extremely fast regardless of size. Keys must be immutable (strings, numbers, tuples), while values can be any type including nested dictionaries.

1

Dictionary Operations

Add or update items by assignment: dict[key] = value. Delete items with del dict[key] or dict.pop(key). Check key existence with key in dict. Get dictionary length with len(dict). These operations enable using dictionaries as flexible data stores.

2

Dictionary Methods

keys() returns all keys, values() returns all values, items() returns key-value pairs as tuples. get(key, default) safely retrieves values with fallback. update(other_dict) merges dictionaries. These methods enable efficient dictionary manipulation.

3

Dictionary Comprehensions

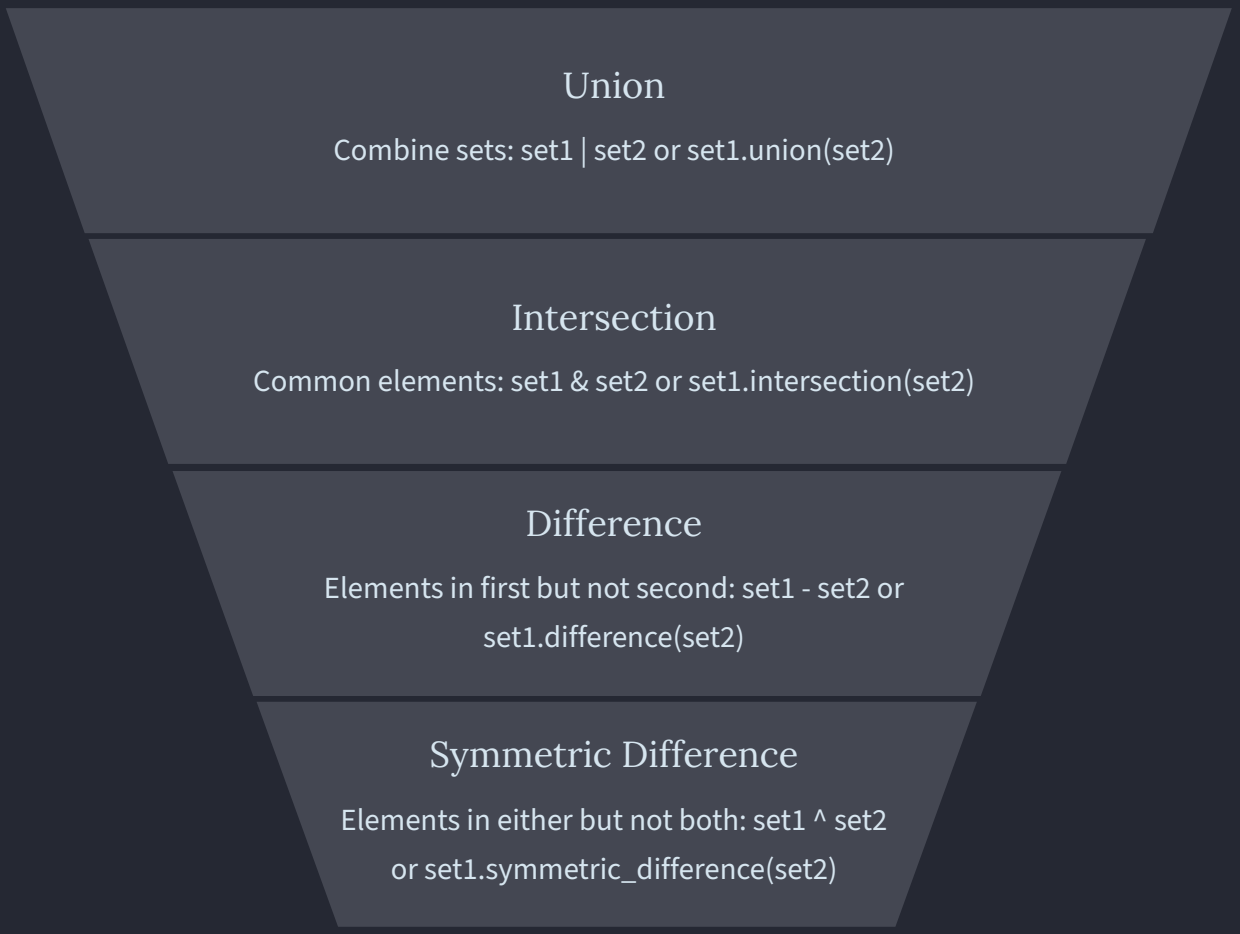
Create dictionaries concisely: {k: v for k, v in pairs if condition}. For example, {x: x**2 for x in range(5)} creates {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}. Comprehensions make dictionary creation elegant and readable.

4

Nested Dictionaries

Dictionaries can contain other dictionaries, enabling complex data structures. For example, users = {'alice': {'age': 30, 'city': 'NYC'}, 'bob': {'age': 25, 'city': 'LA'}}. Access nested values with multiple brackets: users['alice']['city'].

Sets: Creation and Properties use curly braces: {1, 2, 3} or set(). Sets automatically eliminate duplicates and have no order. The three U's define sets: Unordered (no indexing), Unique (no duplicates), Unindexed (no position-based access). Sets excel at membership testing, removing duplicates, and mathematical operations.



Mathematical Set Operations enable elegant solutions to common problems. Find unique items across lists, identify common elements between datasets, or determine differences between configurations. Subset and Superset Checks use issubset() and issuperset() to test containment relationships. Frozen Sets provide immutable sets that can be used as dictionary keys or set elements. Practical Applications include removing duplicates from lists, testing membership efficiently, and implementing mathematical algorithms that rely on set theory.

Advanced Collections & Iterators

Module 5: Specialized Data Structures

Collections Module offers specialized containers beyond built-in types. namedtuple creates tuple subclasses with named fields, improving code readability. Counter counts hashable objects, perfect for frequency analysis. defaultdict provides default values for missing keys, simplifying dictionary operations. deque (double-ended queue) enables efficient appends and pops from both ends, ideal for queues and stacks.

Iterators & Iteration Protocol

Iterators provide a way to access collection elements sequentially without exposing underlying structure. Objects implementing __iter__() and __next__() methods are iterators. Understanding iteration protocol enables creating custom iterators for specialized data sources.

Custom Iterators

Create iterators by implementing __iter__() (returns self) and __next__() (returns next value or raises StopIteration). Custom iterators enable lazy evaluation, processing data on-demand rather than loading everything into memory.

Generators & yield

Generators are functions using yield instead of return, automatically implementing the iterator protocol. Each yield pauses execution, returning a value and resuming when next value is requested. Generators enable infinite sequences and memory-efficient data processing.

Generator Expressions

Similar to list comprehensions but with parentheses: (x**2 for x in range(1000000)). Generator expressions create iterators, not lists, consuming minimal memory regardless of size. Use for large datasets or infinite sequences.

Functions & Scope

Module 6: Code Organization and Reusability

Function Definition and Calling uses the def keyword followed by function name, parameters in parentheses, and a colon. The function body is indented. Call functions by name with arguments in parentheses. Functions can accept any number of parameters and return any type of value, including multiple values as tuples.

01

Positional Arguments

Arguments matched to parameters by position: func(1, 2, 3)

02

Keyword Arguments

Arguments matched by name: func(a=1, b=2, c=3)

03

Default Arguments

Parameters with default values: def func(a, b=10)

04

*args

Arbitrary positional arguments as tuple: def func(*args)

05

**kwargs

Arbitrary keyword arguments as dict: def func(**kwargs)

Function Parameters and Arguments provide flexibility in how functions accept input. Positional Arguments must be provided in order. Keyword Arguments can be provided in any order by specifying parameter names. Default Arguments provide fallback values when arguments aren't supplied. Arbitrary Positional Arguments (*args) collect extra positional arguments into a tuple. Arbitrary Keyword Arguments (**kwargs) collect extra keyword arguments into a dictionary. These features enable creating flexible APIs that adapt to different use cases.

Local Scope

Variables created inside functions exist only within that function. They're destroyed when the function returns, preventing namespace pollution.

Global Scope

Variables created outside functions are accessible everywhere. Use the global keyword to modify global variables from within functions.

Built-in Functions

Python provides 70+ built-in functions like print(), len(), range(), type(), and more. These functions are always available without importing.

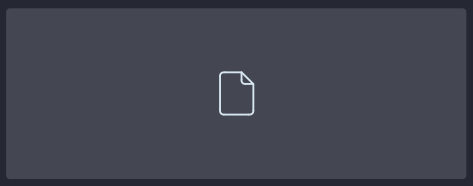
Modules & Packages

Module 7: Code Organization at Scale

Introduction to Modules explains how modules enable code reuse and organization. Any Python file is a module that can be imported into other files. Modules provide namespaces, preventing name conflicts between different parts of your code. Understanding modules is crucial for writing maintainable Python applications.

Built-in Modules

Part of Python standard library, always available



User-Defined Modules

Python files you create in your project

External Packages

Third-party modules installed via pip

Types of Modules include Built-in Modules (part of Python standard library), User-Defined Modules (files you create), and External Modules (third-party packages installed via pip). Each type serves different purposes—built-in modules provide core functionality, user-defined modules organize your code, and external modules add specialized capabilities.

math Module

Mathematical functions: sqrt(), sin(), cos(), pi, e. Essential for scientific computing and calculations.

random Module

Random number generation: random(), randint(), choice(), shuffle(). Useful for simulations and testing.

datetime Module

Date and time manipulation: datetime.now(), timedelta, strftime(). Critical for logging and scheduling.

os Module

Operating system interaction: file operations, directory navigation, environment variables. Essential for DevOps automation.

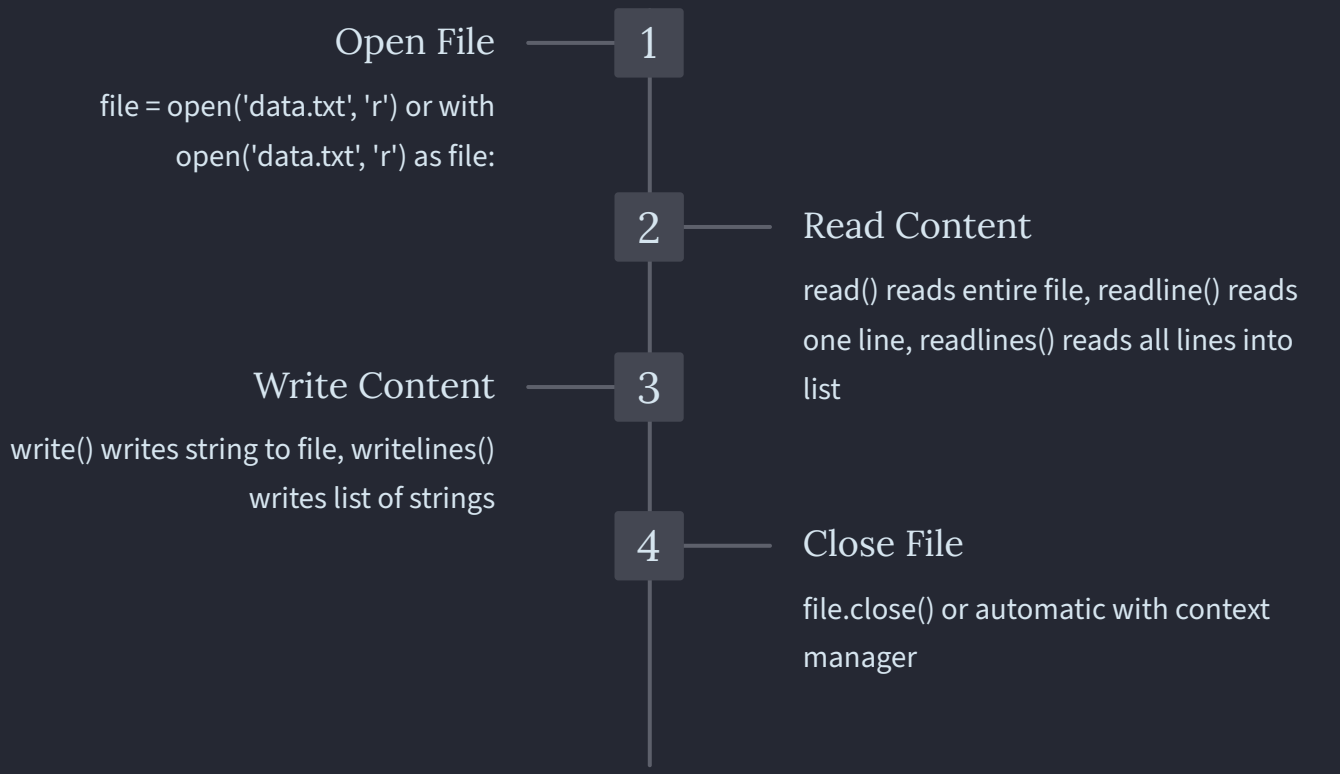
sys Module

System-specific parameters: command-line arguments, exit(), path manipulation. Useful for scripts and tools.

Working with Data Formats

Module 8: Files, CSV, and JSON

File Operations Basics follow CRUD pattern: Create, Read, Update, Delete. The open() function opens files with modes: 'r' for reading, 'w' for writing (overwrites), 'a' for appending, 'r+' for reading and writing. Always close files with close() or use context managers (with statement) for automatic cleanup.



File Path Operations use os.path module for cross-platform path manipulation. join() combines path components, exists() checks if path exists, isfile() and isdir() test path types. Directory/Folder Management uses os and shutil modules. os.mkdir() creates directories, os.listdir() lists contents, shutil.copy() copies files, shutil.rmtree() removes directories recursively.

csv.reader

Reads CSV files row by row as lists. Simple and memory-efficient for processing large CSV files.

csv.writer

Writes lists to CSV files. Handles quoting and escaping automatically for proper CSV format.

csv.DictReader

Reads CSV files as dictionaries using header row as keys. More readable than lists for named columns.

csv.DictWriter

Writes dictionaries to CSV files. Requires specifying fieldnames for column headers.

Advanced Python Concepts

Module 9: Exception Handling, Decorators, and Context Managers

Advanced Python concepts separate competent programmers from experts. Exception handling enables robust error management, decorators add functionality to functions elegantly, and context managers ensure proper resource cleanup. This module teaches you to write production-quality Python code that handles errors gracefully, extends functionality cleanly, and manages resources safely.

Exception Handling Fundamentals enable programs to respond to errors without crashing. Exceptions are events that disrupt normal program flow—file not found, network timeout, invalid input. Python uses try-except blocks to catch and handle exceptions, allowing programs to recover from errors or fail gracefully with meaningful messages.

1

try

Code that might raise exceptions

2

except

Handle specific exceptions

3

else

Runs if no exceptions occurred

4

finally

Always runs for cleanup

Raising Exceptions

Use raise to signal errors: raise ValueError("Invalid input"). Raising exceptions enables functions to communicate errors to callers, who can then handle them appropriately.

Re-raising Exceptions

Catch exceptions, perform cleanup, then re-raise with raise. This enables logging errors while still propagating them to higher-level handlers.

Custom Exceptions

Create exception classes inheriting from Exception. Custom exceptions enable domain-specific error handling and clearer error messages.

Object-Oriented Programming

Module 10: Classes, Objects, and the Four Pillars

Object-Oriented Programming (OOP) organizes code around objects that combine data and behavior. OOP enables building complex systems from simple, reusable components. This comprehensive module covers OOP fundamentals, the four pillars (Encapsulation, Inheritance, Abstraction, Polymorphism), and practical applications. Understanding OOP is essential for working with modern frameworks, libraries, and large codebases.

01

Define Class

Use class keyword to create blueprint

02

Add __init__

Constructor method initializes object state

03

Define Methods

Functions that operate on object data

04

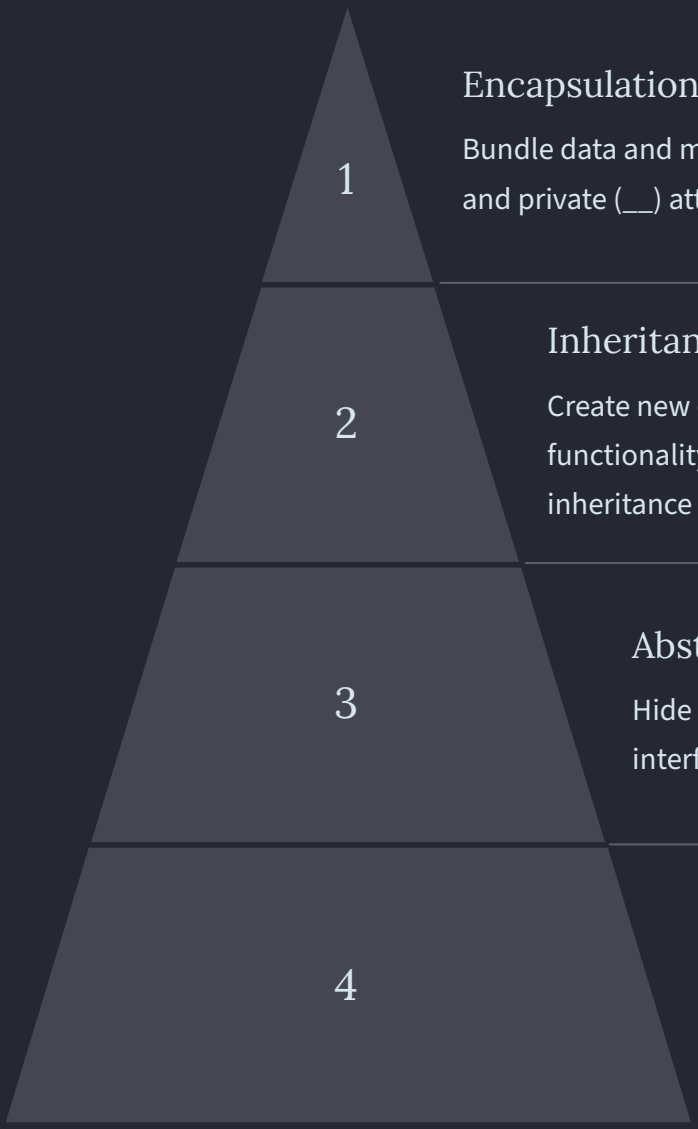
Create Objects

Instantiate class to create objects

05

Use Objects

Call methods and access attributes



Encapsulation

Bundle data and methods, control access with public, protected (_), and private (__) attributes

Inheritance

Create new classes from existing ones, reusing and extending functionality. Supports single, multi-level, and multiple inheritance

Abstraction

Hide complex implementation details, expose simple interfaces. Use abstract base classes to define contracts

Polymorphism

Same interface, different implementations. Method overriding and duck typing enable flexible, extensible code

Four Pillars of OOP provide the conceptual framework. Encapsulation bundles related data and methods, hiding internal details and exposing clean interfaces. Access modifiers (public, protected with single underscore, private with double underscore) control visibility. Inheritance creates hierarchies where child classes inherit parent functionality, enabling code reuse and specialization. Single inheritance has one parent, multi-level inheritance chains multiple parents, multiple inheritance has multiple parents (use carefully due to complexity).

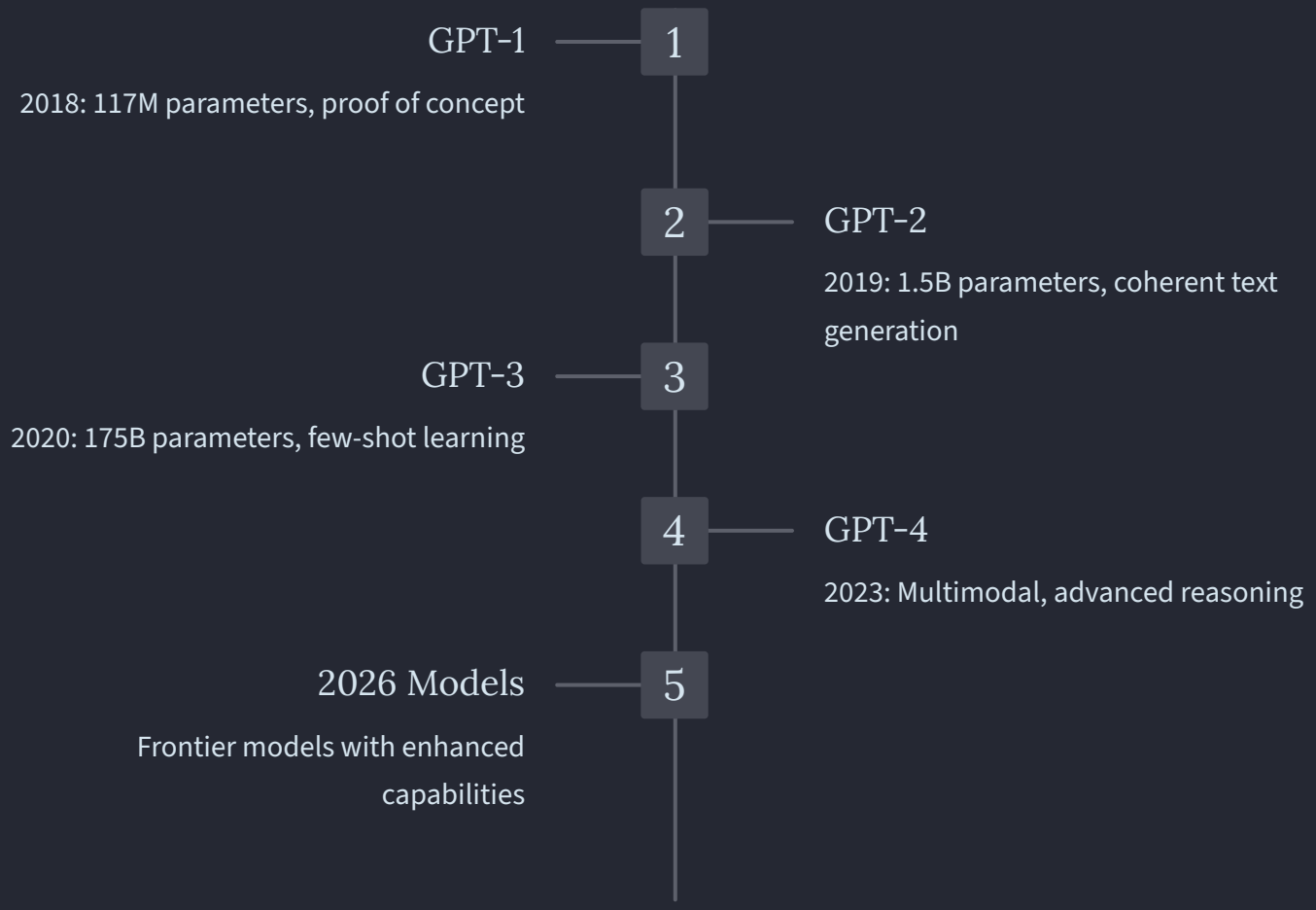
Abstraction hides complexity behind simple interfaces. Abstract classes define contracts that subclasses must implement, using the ABC module and @abstractmethod decorator. Polymorphism enables treating different objects uniformly through shared interfaces. Method overriding allows subclasses to provide specific implementations. Duck typing ("if it walks like a duck and quacks like a duck, it's a duck") enables polymorphism without inheritance.

SECTION 7

Generative AI & Agentic AI

Module 1: Foundations of Generative AI

Generative AI represents a paradigm shift in artificial intelligence, moving from systems that classify and predict to systems that create. Large Language Models (LLMs) can write essays, code, and poetry. Image generators create stunning visuals from text descriptions. This module introduces the foundations of generative AI, covering LLM fundamentals, transformer architecture, and the landscape of frontier models that are reshaping technology and society.



175B

GPT-3 Parameters

Massive scale enables emergent capabilities

200K

Claude Context

Tokens of context for long documents

90%

Cost Reduction

DeepSeek offers comparable quality at fraction of cost

Prompt Engineering & Context Design

Module 2: Maximizing LLM Performance

Advanced prompt engineering techniques go beyond simple instructions. Structure prompts with clear roles, context, instructions, and examples. Use delimiters to separate different sections. Specify output format explicitly. Break complex tasks into steps. These techniques dramatically improve output quality and consistency.

Context Engineering

Provide relevant background information, domain knowledge, and constraints. Context shapes how models interpret instructions and generate responses. Include examples of desired output format and style.

Reasoning Mode Optimization

Enable chain-of-thought reasoning by asking models to "think step by step." This improves accuracy on complex problems by making reasoning explicit. Use reasoning for math, logic, and multi-step tasks.

Reducing Hallucinations

Instruct models to admit uncertainty, cite sources, and stick to provided context. Use retrieval-augmented generation (RAG) to ground responses in factual documents. Validate critical outputs independently.

Zero-shot, few-shot, and chain-of-thought prompting represent different approaches to task specification. Zero-shot provides only instructions, relying on model's pre-training. Few-shot includes examples demonstrating desired behavior, dramatically improving performance on specialized tasks. Chain-of-thought prompting asks models to show reasoning steps, improving accuracy on complex problems requiring multi-step reasoning.

01

Understand Domain

Learn domain terminology, constraints, and success criteria

02

Define Task Clearly

Specify exactly what you want, including format and constraints

03

Provide Context

Include relevant background, examples, and domain knowledge

04

Structure Prompt

Use clear sections, delimiters, and formatting

05

Iterate and Refine

Test prompts, analyze failures, and improve systematically

LLM APIs & LangChain 1.0

Module 3: Building with Language Models

Building production applications with LLMs requires understanding APIs and frameworks that simplify integration. This module covers major LLM APIs (OpenAI, Anthropic, Google, DeepSeek) and LangChain 1.0, the leading framework for building LLM applications. You'll learn to integrate multiple providers, implement middleware for customization, and build cost-optimized pipelines that balance performance and expense.

OpenAI API

GPT models with extensive ecosystem. Offers chat completions, embeddings, and fine-tuning. Strong documentation and community support. Best for general-purpose applications with established patterns.

Anthropic API

Claude models emphasizing safety and long context. Excellent for processing long documents, safety-critical applications, and nuanced reasoning. Longer context windows enable analyzing entire codebases or books.

Google API

Gemini models with multimodal capabilities. Deep integration with Google services. Strong performance on visual tasks and multilingual content. Ideal for applications requiring image understanding or Google ecosystem integration.

DeepSeek API

Competitive performance at significantly lower cost. Excellent for high-volume applications where cost optimization is critical. Comparable quality to major providers for many tasks at fraction of the price.

LangChain 1.0 fundamentals introduce the framework's architecture and core concepts. LangChain provides abstractions for common LLM application patterns—chains for sequential operations, agents for autonomous decision-making, and memory for maintaining conversation context. The framework handles provider differences, enabling switching between models with minimal code changes.

Create_agent abstraction simplifies building autonomous agents that can use tools, make decisions, and accomplish complex tasks. Agents receive goals, plan actions, execute tools, and iterate until goals are achieved. This abstraction handles the complexity of agent loops, tool selection, and error recovery, enabling focus on defining tools and objectives rather than implementation details.

1

Input

User request or data

2

Middleware

Custom processing and routing

3

LLM

Model generates response

4

Post-process

Format and validate output

Middleware systems for customization enable injecting custom logic into LLM pipelines. Middleware can modify prompts, filter outputs, implement caching, or route requests to different models based on content. This flexibility enables building sophisticated applications that adapt behavior based on context, user, or content type.

RAG & Vector Databases

Module 4: Grounding LLMs in Knowledge

Retrieval-Augmented Generation (RAG) addresses LLM limitations by grounding responses in external knowledge. Rather than relying solely on training data, RAG systems retrieve relevant documents and include them in prompts, dramatically improving accuracy and enabling LLMs to work with proprietary or recent information. This module covers vector databases, RAG pipelines, and advanced techniques for building production-quality knowledge systems.

Ingest Documents

Load and chunk documents into manageable pieces

Generate Embeddings

Convert text chunks to vector representations

Generate

LLM generates answer using retrieved context

Store Vectors

Save embeddings in vector database with metadata

Retrieve

Find most similar documents using vector search

Query

User asks question, converted to embedding

Building production RAG pipelines involves multiple stages. Document processing chunks text into optimal sizes (typically 500-1000 tokens), balancing context and retrieval precision. Embedding generation converts chunks to vectors using models like OpenAI's text-embedding-ada-002 or open-source alternatives. Storage in vector databases enables fast similarity search. Query processing converts user questions to embeddings, retrieves relevant chunks, and constructs prompts including retrieved context.

Embedding Strategies

Choose embedding models balancing quality and cost. Consider domain-specific embeddings for specialized content. Experiment with chunk sizes and overlap to optimize retrieval quality.

Hybrid Search

Combine semantic search (vector similarity) with keyword search (BM25) for best results. Hybrid approaches leverage strengths of both methods, improving retrieval accuracy across diverse queries.

Document Processing

Handle various formats (PDF, Word, HTML), extract text cleanly, preserve structure and metadata. Quality document processing directly impacts RAG system performance.

Production Deployment

Module 5: Shipping AI Applications

Building AI applications is one thing; deploying them to production is another. This module covers everything needed to ship reliable, scalable AI applications—from user interfaces with Streamlit and Gradio to enterprise deployment with LangGraph Platform. You'll learn cost optimization, governance, security, monitoring, and scaling strategies that ensure your AI applications perform reliably in production.

01

Develop Locally

Build and test application on development machine

02

Containerize

Package application with dependencies in Docker container

03

Deploy Platform

Deploy to LangGraph Platform or cloud provider

04

Configure Monitoring

Set up logging, metrics, and alerting

05

Optimize Costs

Monitor usage and implement cost controls

LangGraph Platform deployment provides managed infrastructure for LangChain applications. The platform handles scaling, monitoring, and operations, enabling focus on application logic. Deploy with Git integration, configure environment variables, and monitor performance through built-in dashboards. LangGraph Platform simplifies production deployment, reducing operational overhead.

AI Governance

Implement policies for AI usage, content filtering, and human oversight. Document model versions, training data, and decision logic for auditability.

EU AI Act Compliance

Understand risk categories, transparency requirements, and documentation obligations. Implement required safeguards for high-risk AI systems.

API Security

Protect API keys, implement rate limiting, validate inputs, and sanitize outputs. Use authentication and authorization to control access.

Monitoring

Track latency, error rates, token usage, and costs. Implement alerting for anomalies. Log requests and responses for debugging and auditing.

Introduction to Agentic AI

Module 6: Autonomous AI Systems

Agentic AI represents the next evolution of artificial intelligence—systems that can plan, reason, and act autonomously to achieve goals. Unlike traditional AI that responds to prompts, agentic AI takes initiative, uses tools, and adapts strategies based on results. This module introduces agentic AI fundamentals, LangChain 1.0 Agents with middleware, Model Context Protocol (MCP), and enterprise adoption patterns that are transforming how organizations deploy AI.

Goal

User defines objective

Plan

Agent creates action plan

Act

Execute actions using tools

Observe

Evaluate results

Iterate

Adjust plan and continue

Model Context Protocol (MCP) standardizes how agents interact with tools and data sources. MCP defines common interfaces for authentication, data access, and action execution, enabling agents to work with diverse systems through consistent APIs. This standardization accelerates agent development and improves reliability by reducing integration complexity.

Search Tools

Web search, database queries, document retrieval. Enable agents to find information needed for decision-making.

Computation Tools

Calculators, code execution, data analysis. Enable agents to perform calculations and process data.

API Tools

REST APIs, webhooks, service integrations. Enable agents to interact with external systems and services.

Communication Tools

Email, messaging, notifications. Enable agents to communicate with humans and other systems.

LangGraph 1.0 Fundamentals

Module 7: Building Stateful AI Workflows

LangGraph 1.0 provides a framework for building stateful, graph-based AI workflows. Unlike simple chains that execute linearly, LangGraph enables complex workflows with conditional routing, parallel execution, and iterative refinement. This module covers LangGraph architecture, state management, and production features that enable building sophisticated AI applications with reliability and maintainability.

State Management

LangGraph maintains state throughout workflow execution, enabling nodes to access and modify shared data. State is typed and validated, preventing errors from invalid data. State management enables building workflows that accumulate information, make decisions based on history, and maintain context across multiple steps.

Graph-Based Logic

Define workflows as graphs with nodes (operations) and edges (transitions). Conditional edges enable branching based on state. Parallel edges enable concurrent execution. Loops enable iterative refinement. This flexibility enables expressing complex business logic clearly and maintainably.

Node Caching

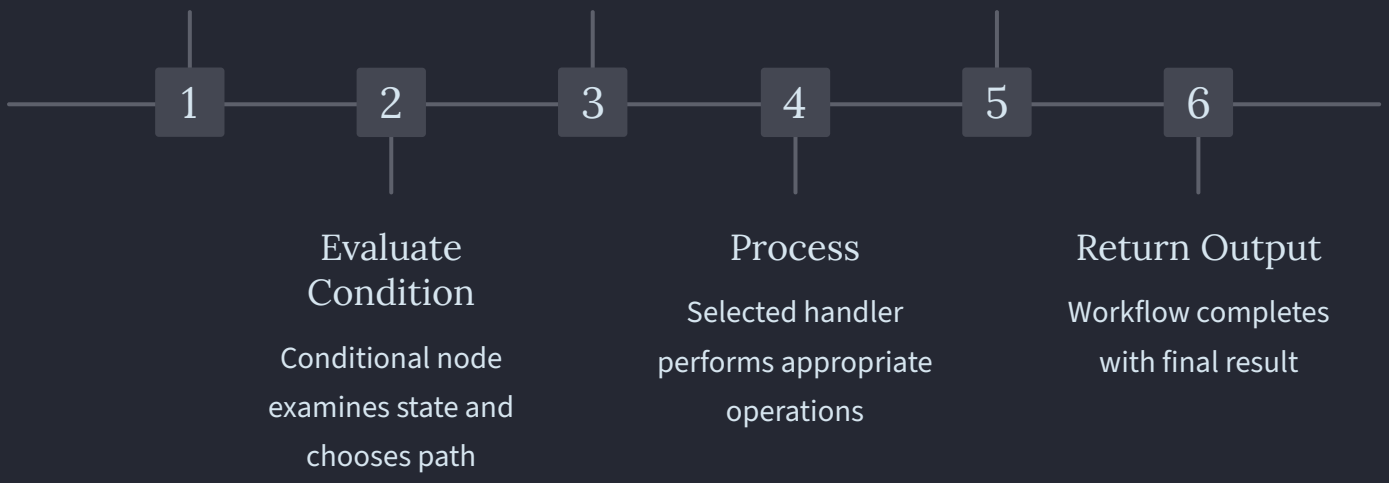
Cache node results during development to avoid redundant API calls. Caching accelerates iteration by reusing expensive operations. In production, caching can reduce costs by avoiding duplicate work. Configure caching per node based on operation characteristics.

Pre/Post Hooks

Implement guardrails using hooks that run before or after nodes. Pre-hooks validate inputs, check permissions, or modify state. Post-hooks validate outputs, log results, or trigger notifications. Hooks enable implementing cross-cutting concerns without cluttering node logic.

Advanced Workflow Patterns

Module 8: Sophisticated AI Orchestration



Essay Evaluation

Analyze essays across multiple dimensions (grammar, structure, argumentation), provide detailed feedback, and suggest improvements. Parallel evaluation of different aspects reduces latency.

Feedback Routing

Classify customer feedback by topic and sentiment, route to appropriate teams, and prioritize based on urgency. Conditional routing ensures feedback reaches the right people.

Approval Workflows

Multi-stage approval processes with conditional routing based on amount, type, or risk. Parallel approvals from multiple stakeholders reduce cycle time.

Content Generation

Generate content, evaluate quality against standards, and iterate until quality gates are met. Quality-gated workflows ensure consistent output quality.

Production Agentic Systems

Module 9 & 10: Enterprise-Grade AI Agents

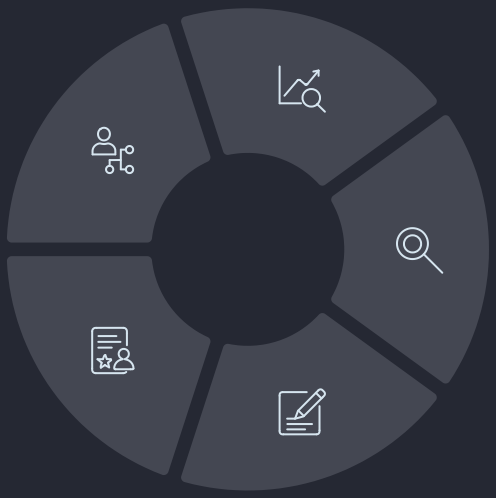
This final module covers LangGraph Platform deployment, multi-agent system design, Google A2A Protocol for agent communication, LangSmith observability, MCP security, and comprehensive guardrails that ensure agents operate safely and reliably in production environments.

Coordinator Agent

Orchestrates other agents, delegates tasks, and combines results

Review Agent

Evaluates quality and provides feedback



Research Agent

Gathers information from various sources

Analysis Agent

Processes data and generates insights

Writing Agent

Creates content based on research and analysis

LangSmith Observability

Track agent decisions, tool usage, and performance. LangSmith provides detailed traces showing how agents reason and act, enabling debugging and optimization. Observability is critical for understanding agent behavior and building trust.

MCP Security Model

Implement authentication, authorization, and encryption for agent-tool interactions. MCP security ensures agents can only access authorized resources and that communications are protected from interception.

Prompt Injection Prevention

Detect and block attempts to manipulate agent behavior through crafted inputs. Implement input validation, output sanitization, and behavioral monitoring to prevent prompt injection attacks.

Congratulations! You've completed this comprehensive DevOps and AI training program. You now possess the skills to build, deploy, and maintain modern applications using cutting-edge DevOps practices and AI technologies. Continue learning, practicing, and building—the future of technology is in your hands.