Cloud Computing & DevOps Documentation

📌 Module 1: Introduction to Cloud Computing

🔹 What is Cloud Computing?

Cloud computing is the **delivery of computing services** (servers, storage, databases, networking, software, and more) over the internet, offering **on-demand availability, scalability, and cost efficiency**.

🔹 Key Features of Cloud Computing:

✅ **On-Demand Self-Service** – Users can provision resources as needed.
✅ **Broad Network Access** – Accessible from anywhere with internet connectivity.
✅ **Resource Pooling** – Providers serve multiple customers with shared resources.
✅ **Rapid Elasticity** – Can scale up/down based on demand.
✅ **Measured Service** – Pay-as-you-go pricing model.

📌 Module 2: Types of Cloud Providers & Cloud Models

🔹 Cloud Providers:

- **Amazon Web Services (AWS)** – Market leader with IaaS, PaaS, and SaaS offerings.
- **Microsoft Azure** – Strong enterprise and hybrid cloud capabilities.
- **Google Cloud Platform (GCP)** – Specializes in AI/ML, Big Data, and Kubernetes.
- **IBM Cloud, Oracle Cloud, Alibaba Cloud** – Niche enterprise solutions.

🔹 Types of Cloud Models:

Cloud Type	**Description**	**Use Case**
Public Cloud	Infrastructure owned by cloud providers, shared among multiple organizations.	Hosting websites, scalable applications, startups.

Private Cloud	Exclusive infrastructure for a single organization, better control & security.	Banking, government, healthcare, and regulated industries.
Hybrid Cloud	Combination of public and private clouds.	Enterprise with both on-premise & cloud-based apps.
Multi-Cloud	Using multiple cloud providers for different workloads.	Avoiding vendor lock-in, high availability.

📌 Module 3: Deployment of Projects on Cloud with DevOps

💠 DevOps in Cloud Computing:

- **DevOps** integrates **development (Dev) and operations (Ops)** to streamline **CI/CD pipelines, automation, and monitoring**.
- Cloud platforms offer **DevOps tools** like AWS CodePipeline, Azure DevOps, Google Cloud Build.

💠 Deployment Steps:

1️⃣ **Source Code Management:** Store code in GitHub/GitLab/Bitbucket.
2️⃣ **Continuous Integration (CI):** Use Jenkins/GitHub Actions to automate builds.
3️⃣ **Containerization:** Use Docker/Kubernetes to package applications.
4️⃣ **Infrastructure as Code (IaC):** Use Terraform, CloudFormation for provisioning.
5️⃣ **Deployment:** Deploy using cloud-native services like AWS ECS, Azure App Service.
6️⃣ **Monitoring & Logging:** Use CloudWatch, Azure Monitor, Prometheus.

📌 Module 4: Centralized vs. Distributed Version Control Systems

Feature	Centralized VCS (CVCS)	Distributed VCS (DVCS)
Examples	SVN, Perforce	Git, Mercurial

Repository Type	Single central repository	Every user has a local copy
Speed	Slower (requires network access)	Faster (local commits possible)
Availability	If the central server is down, no commits	Work can continue offline
Branching & Merging	Complex	Easier & more flexible

💡 **Git is the most widely used DVCS**, preferred for cloud-native and DevOps workflows.

📌 Module 5: Git Basics - Branching, Merging, Remote Repositories

💠 Common Git Commands:

Command	**Description**
`git init`	Initialize a new repository
`git clone <repo_url>`	Clone an existing repository
`git add .`	Stage all changes for commit
`git commit -m "message"`	Commit staged changes
`git push origin main`	Push commits to a remote branch

`git pull origin main`	Pull latest changes from the remote branch
`git merge <branch_name>`	Merge another branch into the current branch
`git branch -d <branch_name>`	Delete a local branch

◈ Best Practices for Git Usage:

☑ Use **feature branches** (`feature/login-ui`) instead of committing directly to `main`.
☑ Commit **small, meaningful changes** with proper messages.
☑ Always pull **latest changes** before pushing (`git pull --rebase`).
☑ Regularly review **merge conflicts** and resolve them efficiently.