Module 1: Azure DevOps Overview

1. Introduction to Azure DevOps

Azure DevOps is Microsoft’s fully managed DevOps platform that provides a **complete toolchain for modern software development**, covering planning, coding, building, testing, releasing, and monitoring.

It helps organizations adopt a true **DevOps culture** by providing:

- **Automation** through CI/CD
- **Collaboration** via Repos and Boards
- **Quality assurance** via Test Plans
- **Continuous delivery** through release pipelines
- **Reusable components** via Artifacts
- **Visibility & traceability** across all stages

Azure DevOps supports any language, any platform, and integrates seamlessly with Azure Cloud, on-prem workloads, and hybrid architectures.

---

2. Why Azure DevOps? (Enterprise Perspective)

2.1 Benefits

- **End-to-end DevOps lifecycle** in a single platform
- **Enterprise-ready** security, compliance, and governance
- **Highly scalable** (supports small teams to global enterprises)
- **Marketplace Integrations** (Slack, SonarQube, JIRA, Docker, etc.)
- **Multi-cloud deployments** (Azure, AWS, GCP, Kubernetes, VMs)

2.2 Common Use Cases

- Microservices CI/CD pipeline
- Terraform/ARM/Bicep IaC deployment
- Automated testing workflow
- Agile project management & sprint planning
- Artifact and dependency management
- Enterprise-level testing (manual + automated)

---

3. Azure DevOps vs. GitHub Actions vs. Jenkins

3.1 Summary Table

Capability	**Azure DevOps**	**GitHub Actions**	**Jenkins**
Version Control	Git + TFVC	Git (GitHub-only)	Git, SVN, Mercurial
CI/CD	Pipelines (YAML/Classic)	Workflows (YAML)	Freestyle + Declarative
Project Management	Full agile suite (Boards)	Lightweight Projects	No native PM
Testing	Test Plans	No enterprise suite	Plugins
Packages	Azure Artifacts	GitHub Packages	Plugins
Hosting	Cloud + Self-host	Cloud + Self-host	Self-host only
Best For	Enterprises	GitHub-native teams	Custom open-source CI/CD

3.2 Key Differences

Azure DevOps

- Suitable for large enterprises
- Traceability from idea → deploy
- Strong governance & compliance
- Supports non-GitHub workflows
- Deep Azure & cloud integrations

GitHub Actions

- Best for open-source and GitHub-native workflows
- Very simple YAML workflows
- Easy community collaboration

Jenkins

- Extremely flexible
- Large plugin ecosystem
- Operational overhead (maintenance, scaling, plugin upgrades)

---

4. Azure DevOps Core Services

4.1 Azure Repos

Azure Repos provides **centralized code management** with:

Features

- Git repositories
- Branching strategies (GitFlow, Trunk-based)
- Pull Requests with policies
- Code reviews & discussions
- Build validation (CI as gate)
- Branch security rules
- Commit history & audit logs

Branch Policies

Recommended:

- Require minimum reviewers (2)
- Enforce linked work items
- Require successful build (CI)
- Limit who can approve own PR

4.2 Azure Pipelines

Azure Pipelines enables **CI/CD automation** with multi-agent, multi-stage YAML or Classic pipelines.

Capabilities

- Build automation (CI)
- Testing automation (unit/functional)
- Packaging & artifact creation
- CD deployments to:
  - Azure App Services
  - Kubernetes (AKS/EKS/GKE)
  - VMs (Windows/Linux)
  - Containers

    - On-prem servers

Pipeline Types

1. **Classic Pipelines**
    - Drag-drop, UI-based
    - Faster onboarding for beginners
2. **YAML Pipelines**
    - “Pipeline-as-code”
    - Version-controlled
    - Supports multi-stage deployments

Pipeline Structure (YAML)

- Trigger
- Stages
- Jobs
- Steps
- Tasks
- Agents

4.3 Azure Boards

Boards is a **complete Agile project management solution**.

Hierarchy

- Epics
- Features
- User Stories
- Tasks
- Bugs

Capabilities

- Backlogs
- Sprint planning
- Kanban boards
- Capacity planning
- Dashboards

- End-to-end traceability
- Analytics (velocity, burndown, cycle time)

Traceability

A key enterprise strength:
Story → Code → Commit → Build → Release → Deployment → Test Case → Defect

---

4.4 Azure Artifacts

A **secure package hosting and management service**.

Supported Package Types

- npm
- Maven
- NuGet
- Python
- Universal Packages

Benefits

- Internal secure registry
- Version management
- Reusability across microservices
- Supports retention & cleaning rules

4.5 Azure Test Plans

An **enterprise-grade testing solution** for manual, exploratory, regression, and automated testing lifecycle.

Features

- Manual Test Suites
- Automated Test Integration (via Pipelines)
- Exploratory Testing Session
- Bug tracking with evidence
- Traceability with Boards & Pipelines

Benefits

- Ensures quality across releases
- Helps regulated industries (BFSI, Pharma, Insurance)

- Centralized testing dashboard

---

5. DevOps Lifecycle

The DevOps lifecycle emphasizes **continuity, automation, and feedback**.

Diagram – DevOps Lifecycle

```
PLAN → CODE → BUILD → TEST → RELEASE → DEPLOY → OPERATE → MONITOR
     ↑                          ↓
     └──────────── Continuous Feedback ──────────┘
```

5.1 Phase-by-Phase Description

PLAN

- Requirement gathering
- Sprint planning
- Roadmap creation
- Work item creation (Boards)

CODE

- Develop features
- Follow coding standards
- Use branching (GitFlow/Trunk)
- Conduct code review via pull requests

BUILD (CI)

- Compile code
- Run static analysis
- Run unit tests
- Generate build artifacts

TEST

- Integration tests
- Regression tests
- Security scans (SAST/DAST)
- Performance tests

RELEASE

- Release pipelines (CD)
- Approval gates
- Automated versioning
- Blue/Green & Canary strategies

DEPLOY

- Deploy across environments:
  - Dev → QA → UAT → Prod
- Use IaC (Terraform, ARM, Bicep)
- Use Key Vault for secrets

OPERATE

- Infrastructure operations
- Configuration management
- Auto-scaling & health monitoring

MONITOR

- Application telemetry
- Logs, metrics, traces
- Alerting
- Feedback to the product backlog

---

6. Azure DevOps Hierarchy

Azure DevOps follows a **top-down structured hierarchy**.

Diagram – Azure DevOps Hierarchy

```
┌──────────────────────────────┐
│    Azure DevOps Organization    │
└───────────────┬───────────────┘
                ↓
     ┌────────────┐
     │  Project  │
     └─────┬─────┘
```

```
 ┌────────────────────┼────────────────────────┐
 ↓            ↓              ↓
┌──────────┐   ┌────────────┐    ┌────────────┐
│ Repos    │   │ Pipelines │     │ Boards    │
└──────────┘   └────────────┘    └────────────┘
          ↓
       ┌────────────┐
       │ Artifacts  │
       └────────────┘
          ↓
       ┌────────────┐
       │ Test Plans │
       └────────────┘
```

6.1 Organization

- Highest-level container
- Can host multiple projects
- Identity source controlled by Azure AD
- Used for company-level governance

6.2 Project

- Logical boundary for a product/team
- Contains:
  - Repos
  - Pipelines
  - Boards
  - Test Plans
  - Artifacts

You can have multiple microservices within the same project or separate projects.

6.3 Azure Repos

- Multiple Git repositories
- Access controlled at repo or branch level

6.4 Azure Pipelines

- You can create:
  - One pipeline per repo
  - Multiple pipelines per repo
  - Multi-service pipelines

6.5 Azure Boards

- One project can have:
  - Multiple teams
  - Separate backlogs
  - Shared or separate sprints

---

7. End-to-End Azure DevOps Workflow Diagram

Developer Pushes Code → Azure Repos

↓

CI Pipeline Triggers (Build + Unit Tests + SAST)

↓

Artifact Published (Packages/Images)

↓

CD Pipeline (Deploy to Dev/QA/UAT/Prod)

↓

App Insights / Monitor

↓

Bugs/Roadmap updated in Azure Boards

---

8. Best Practices in Azure DevOps

Source Control

- Use trunk-based development for faster releases
- Enable PR validation
- Protect main branch

Pipelines

- Prefer YAML pipelines
- Store YAML in same repo
- Use templates for reusability
- Enable parallel agents

Testing

- Integrate automated testing
- Run unit + integration + security tests in CI

Security

- Store secrets in Key Vault
- Enable RBAC & least privilege
- Implement OWASP-based scanning

Release Management

- Use multi-stage YAML
- Use environment approvals
- Canary/Blue-Green deployments