# AZ-2001: Implement Security Through a Pipeline Using Azure DevOps

**Modern software delivery pipelines face a dual challenge: delivering features faster while protecting applications and data against increasingly sophisticated threats. Traditional security practices that operate only at the end of the development lifecycle are no longer sufficient. To meet today's demands, organizations must adopt a "shift left" security strategy, embedding controls into every stage of the DevOps process. Azure DevOps provides a powerful platform for this approach. By combining project structure, secure pipeline resources, strong identity management, permission validation, and reusable templates, teams can build pipelines that are both agile and secure.**

## 0. Introduction

- Welcome and agenda.
- Why security must be built into DevOps pipelines
- Learning goals: securing repos, resources, identities, permissions, and templates.

## 1. Configure Project and Repository Structure

- Importance of structuring projects for security.
- Single vs. multiple projects: trade-offs in visibility, autonomy, and alignment.
- Organizing and securing repositories: controlling branch policies, handling forks, protecting against external contributions.
- Security groups: default, custom, and team groups.
- Project vs. repo permissions
- Demo idea: Configure project-level vs. repo-level permissions.

## 2. Configure Secure Access to Pipeline Resources

- What are Secure Files and why they matter.
- Service connections: enabling secure automation and deployments.
- Managing environments (dev, test, prod): traceability and zero-downtime strategies.
- Accessing Azure Repos and Azure Artifacts from pipelines securely.
- Secret variables and variable groups.

- Demo idea: Add a Secure File and consume it in a YAML pipeline.

## 3. Manage Identity for Projects, Pipelines, and Agents

- Microsoft-hosted vs. self-hosted agents.
- Agent identities: controlling what agents can access, auditing, and resource allocation.
- Service connection scopes: least privilege principle, review/update, monitor usage.
- Managed identities in Azure DevOps: benefits over service principals.
- Demo idea: Convert a service connection to use a managed identity.

## 4. Configure and Validate Permissions

- Why validating permissions is essential.
- Configuring user, pipeline, and approval permissions.
- Approvals and branch checks for controlled releases.
- Auditing with Azure DevOps logs: monitoring for unauthorized changes.
- Demo idea: Run a permissions audit and test access.

## 5. Extend a Pipeline with Templates

- Security advantages of YAML templates.
- Types of templates: stage, job, step, variable.
- Rewriting deployment pipelines to consume templates.
- Tokenization with Key Vault: best practices
- Restricting agent logging: secret masking, issecret parameter, auditing.
- Conditionally removing risky script tasks with YAML expressions.
- Demo idea: Build a nested template and deploy with tokenized secrets.

## 6. Wrap-Up & Next Steps

- Recap of modules
- Q&A.