# Definition of Done

The Definition of Done (DoD) is a crucial agreement in agile development that establishes clear, shared criteria for when a work item is considered complete. It serves several key purposes:

1. **Quality Assurance**: Creates a consistent standard of quality by defining mandatory checkpoints that all work must meet before being accepted.
2. **Alignment**: Ensures all team members and stakeholders have the same understanding of what “complete” means, reducing misunderstandings and rework.
3. **Predictability**: Helps teams make reliable commitments by having clear completion criteria that inform estimation and planning.
4. **Risk Reduction**: Prevents incomplete or substandard work from progressing through the development pipeline by establishing minimum acceptable standards.
5. **Transparency**: Makes the development process more visible and measurable by providing concrete criteria against which progress can be assessed.

The DoD acts as a contract between the development team and stakeholders, protecting both parties by clearly defining expectations and requirements for completion. This shared understanding is essential for maintaining velocity, quality, and trust in the agile development process.

## ADO Features

Managing features and user stories in Azure DevOps (ADO) involves a structured and collaborative approach to ensure clarity, alignment, and efficient delivery within agile sprints. The following outlines the detailed process:

### Defining Features and User Stories

* **Features**: Represent high-level capabilities or functionalities that deliver value to end-users or stakeholders. Features are typically broader in scope and may span multiple sprints.
* **User Stories**: Smaller, actionable units of work derived from features. Each user story clearly describes a specific requirement from the user’s perspective, typically following the format: “As a [user role], I want [functionality], so that [benefit].”

### Creating and Organizing Work Items in ADO

* **Work Item Creation**: Product Owners or Business Analysts create features and user stories directly in ADO, clearly defining acceptance criteria, priority, and business value.
* **Hierarchy and Relationships**: User stories are linked to their parent features using ADO’s built-in hierarchical relationships. This ensures traceability and clear visibility into how individual stories contribute to overall feature completion.

### Prioritization and Backlog Management

* **Backlog Refinement**: Regular backlog refinement sessions are conducted to review, prioritize, and clarify features and user stories. The Product Owner, Scrum Master, and development team collaborate to ensure the backlog accurately reflects business priorities and technical feasibility.
* **Priority Assignment**: Features and user stories are prioritized based on business value, dependencies, risk, and stakeholder input. ADO’s backlog view allows easy drag-and-drop prioritization and clear visualization of the order of work.

### Sprint Planning and Grouping into Collections

* **Sprint Planning Meetings**: At the start of each sprint, the agile team conducts sprint planning sessions. During these sessions, the team selects a collection of prioritized user stories from the backlog that can realistically be completed within the sprint duration.
* **Grouping into Collections**: Selected user stories are grouped into a sprint backlog within ADO. This collection represents the team’s commitment for the sprint and provides a clear scope of work for the upcoming development cycle.

### Handing Over to Engineering for Development

* **Sprint Backlog Commitment**: Once the sprint backlog is finalized, it is formally handed over to the engineering team. Developers pick user stories from the sprint backlog, assign them to themselves, and begin development activities.
* **Tracking and Transparency**: Throughout the sprint, progress on user stories is tracked using ADO boards (Kanban or Scrum boards). This provides real-time visibility into the status of each story, enabling effective collaboration and timely identification of any impediments.

### Completion and Validation

* **Definition of Done (DoD)**: Each user story must meet the agreed-upon Definition of Done criteria before being marked as complete. This typically includes code completion, peer reviews, automated tests, documentation, and stakeholder acceptance.
* **Sprint Review and Retrospective**: At the end of the sprint, completed user stories are demonstrated to stakeholders during the sprint review. Feedback is collected, and any necessary adjustments are made. The retrospective meeting follows, allowing the team to reflect on the sprint and identify opportunities for continuous improvement.

By following this structured approach in Azure DevOps, teams ensure clear communication, alignment on priorities, efficient development cycles, and consistent delivery of high-quality features and user stories.

## Completion of Feature or Bug

Once an Engineering member has delivered a feature or a bug into the development environment the version of the modules that will be effected by the feature or bug will be updated in gitops repo for the development environment for the modules effected. This will be done in the form of a pull request following the same proceedure and the creation of a branch to support the update of a feature. This pull request will be applied to a release branch that will be created at the begining of sprint that will accept all the changes the engineering team is completing. The pull request will focus on updates to the *qa* folder structure.

### Updating QA Environment in GitOps Repo

* **Updating QA Folder**: Once the feature or bug is completed and the version information is updated in ADO, the corresponding QA folder in the GitOps repo is updated. This triggers ArgoCD to automatically detect the changes and initiate a deployment of the updated code into the QA environment.
* **Deployment Verification**: The updated code is deployed into the QA environment, and the QA team verifies the functionality and performance of the new feature or bug fix.
* **Feedback and Iteration**: If any issues are identified during the QA testing, the development team makes the necessary adjustments, and the process is repeated until the QA team approves the deployment.

The feature or user story or bug in ADO will now be maked as complete and the version information updated in ADO. Once the whole release is complete it should contain a list of all the features and the version of modules that are effected.

### Moving Release to UAT

* **Acceptable User Testing (UAT) Completion**: Once the combined release has passed acceptable user testing in the QA environment, it is ready to be moved to the UAT environment.
* **Moving to UAT Folder in GitOps Repo**: This is achieved by moving the contents of the QA folder to the UAT folder in the GitOps repo.
* **Triggering Full Deployment**: This action triggers a full deployment of the release branch, ensuring that the UAT environment is updated with the latest features and bug fixes.

### Moving Release to Production

* **UAT Acceptance**: Once the release has passed UAT, it is ready to be moved to the production environment.
* **Moving to Production Folder in GitOps Repo**: This is achieved by moving the contents of the UAT folder to the production folder in the GitOps repo.
* **Triggering Full Deployment**: This action triggers a full deployment of the production branch, ensuring that the production environment is updated with the latest features and bug fixes.

### Future Automated Testing and Release Management

* **Test Suites**: Automated test suites are created for various levels of testing, including unit tests, integration tests, API tests, performance tests, and end-to-end user journey testing. These test suites are integrated into the CI/CD pipeline and are triggered automatically upon code changes.
* **Quality and Success**: The results of the automated test suites now trigger a level of quality and success automatically. If the tests fail, the code changes are not merged, ensuring that only high-quality code is promoted.
* **Automated Release Cycle**: The updates to the GitOps repo, triggered by the completion of the automated test suites, can now be automated to move features and bugs through the release cycle automatically. This ensures a streamlined and efficient release management process, reducing manual intervention and human error.