

Azure DevOps - Application Lifecycle management in the cloud.

You may be familiar with Azure DevOps and the basics, if not, this article may help you learn and identify the most core aspects of the interface and software development and its dynamic set of tools and accessories which will help you easily produce software applications.

Briefing >>ALM(Application Lifecycle Management) for Azure DevOps.

The **A**pplication **L**evel **M**anagement system in Azure DevOps is a dynamic ecosystem for lifecycle planning and operations management systems for developing and deploying software. The interface allows cross-functional operations and understanding between the teams.

It improves the process of software planning and development by presenting an entire range of tools and frameworks to supervise, plan, test, and monitor the development of your desired app. It also supports innovative functionalities for teams to collaborate and control the software lifecycle from initiation to deployment and launch.

The solution system has every type of functionality within it, ranging from source code control build tests, agile building and automated test releases, regular integration, comprehensive monitoring, and delivery.

The different test frameworks that Microsoft Azure DevOps supports are jasmine, Xunit, Mochat, selenium which will test the project and help with the development. ALM also refers to the different changes within the software and how it has been documented and tracked.

“As you develop the software in Azure DevOps, you can send and receive constant feedback and suggestions with your software development team or network, it will also help you collectively work and plan your next step or set of changes in the software development process, iteratively.

Every iteration in a software development project will give you better insights and ideas for improving your software and the knowledge to fix the bugs in the code To understand things better, let us map out a clear and accurate thought process involved in an ALM lifecycle.”

The 4 phases of the ALM software development are:

1. Plan

2. Develop + Test
3. Release
4. Monitor + Learn

While the above 4 phases of software development are constant and systematic, changes that occur in the software while planning, developing, testing, or release and monitoring stages are immediately reported to members associated with the project.

During the release phase, however, some additional steps like the **automated functional testing phase** and the **cloud testing phase are implemented**. The above two steps need not be repeated all the time in the software development lifecycle.

As you read further, you will learn all the important facets and functionalities of Azure DevOps lifecycle management.

What is Azure DevOps?

If you require an accurate definition of Azure DevOps, then here it is⇒

Azure DevOps is a (Software as a service)SAAS suite from Microsoft, it implements end-to-end development and deployment programming tools or (DevOps toolchain) for developing your software product/project from its beginning to the ultimate release of the product.

Formerly known around the world as VSTS(Visual Studio Team System), it retained many of its original functionalities like planning the projects, management of source codes from repositories like TFVC, Subversion, Git, Git Hub, utilizing Ci/CD pipeline for deploying the code into the cloud or on-ground resources.

Besides that, it expedites software planning and development by facilitating a collaborative platform to bring the business users, developers, project managers, and test engineers under one umbrella.

They can observe, track, manage and approve the functions and changes within the software in real-time. It greatly expedites the software development and hence quickly prepares the product for manufacturing and shipment.

What do Azure DevOps Services Contain?

Azure DevOps contains a comprehensive set of services necessary for the outright life-cycle development of the software and the release.

- **Azure Board:** A platform that supports features like active planning, work item tracking, visualization of software development stages by members, and real-time reporting tools.

- **Azure Pipelines:** A software coding and building pipeline tool used for continuous integration and code delivery into a cloud-agnostic CI/CD platform with support for containers and Kubernetes.
- **Azure Repos** provides cloud-hosted private Git, GitHub repositories for coding and software development. You can also use repos services to pull requests for changes and a collaborative development environment.
- **Azure Artifacts:** to implement integrated packaging management with packages like Maven, npm, Python, and NuGet packages that utilize public or private sources.
- **Azure Test Plans:** facilitating integrated planned and exploratory testing solutions for running tests, validation, and verification of desired functionalities, it also provides options to observe and raise logs if any defects or failures are detected in test scenarios.

Azure DevOps can also be used to organize third-party tools.

How to Administer Azure DevOps:

As we now have an understanding of the basics of Azure DevOps and its core functionalities, let us move into the section about administering or working with the suite.

Unlike traditional software development suites, Azure DevOps supports a culture of including project managers, contributors, and developers into projects and integrate their efforts and ideas to develop software in real-time.

What you get is a fast-paced environment to develop software projects compared to the traditional approach. It also promotes best practices in the organization and creates better products at an improved rate.

The software suite also has dedicated cloud services for work, along with the option to work on-premises using the Azure DevOps server.

You can access basic features and other integrated features using a web browser or IDE client. According to one's business needs any of the project development partners can utilize one or multiple kinds of standalone services that the software solutions provide.

Initial Steps of working in Azure DevOps

- **Create Your Organisation** – This is basically creating your Azure DevOps account name and inviting others and giving them access to your organization. Typically you will have one Azure DevOps account and manage multiple projects within.
- **Start Projects** – The organization is split into many multiple projects. This will provide you with the power to manage access control and then split the code and pipelines from each other.

- **Choosing the Repo** – Git or Team Foundation Services Source Control to basically what functionalities are needed for the project.

How to Manage your projects in AZURE DevOps?

Azure DevOps supports a centralized and seamless project management system because of the integrated tools in the platform. Let us explore the project management features included in the Azure DevOps Lifecycle management.

Project management features are classified into the following:

People:

Azure DevOps can classify the different people involved in the project as workgroups, teams, or individuals following the traditional approach or an agile approach. All teams can be included in the interface by linking to the Azure Directory(AD), on-prem(AD), and even the option to invite guests to work on parts or instances of the project.

Every person involved is subject to access control, you have complete control over what they can view and what they don't need. Any of the probable users can be added to the teams where they can view, work access code, and team up to work in tandem.

Users:

Users are persons added by the administrator of the organization. Once added, any user can set up their profile by adding an image of theirs and writing their personal details.

Azure AD/Guests:

You can invite others from different organizations to work on your project or join the project. Such users are called guests. These guests can either be any consultant working on your projects or people you need to share your work with. Entering the person's email address and select the team.

Groups & permissions:

Primary access control mechanism furnished by Azure DevOps. Several built-in groups on the platform can deliver users access to many features. The organization settings page has various settings and levels of permissions to give to certain people or groups in the project development.

Adding people to these groups and assigning these groups or individuals to one or more projects can help your project accordingly. The various levels of permissions will define the level of access each group or individual has in the project.

Teams:

Teams are internal classifications of people or groups in the project who are given certain powers within the project, but it does not mean every team will have the same rights as others.

The topmost authority will always be the administrator who will decide which individuals will see what. For example, a particular team will only have scope to see the portfolio, area, projector. The test team will only see bugs, stories, and tasks for them on the azure board. Meanwhile, the project manager can only view portfolios and epics bombarded with details.

Processes: Projects may vary across organizations according to the processes involved in them. Azure DevOps has provided a selection of templates for the azure boards to be customized for different needs.

Process types: The process type boards are basically common fields and structures that can help you manage and work within different process types. They include all kinds of functionalities like agile, basic complex, SCRUM. CMMI. From these different process types, you can create inherited process types or define these processes by adding fields or names that you like.

Back Logs: Backlogs are a section within the platform where you can manage all work items. They include product, feature planning, for working in the organization. They are usually classified into different levels, each of the backlogs depends on the different backlogs selected. There are mainly 3 kinds of backlogs.

- **Portfolio backlogs:** contains epics and features for higher-level authority.
- **Requirment Backlogs:** This Backlog describes issues, product backlog items, and user stories.
- **Iteration Backlog:** task-level backlog for each level of time-scoped iteration

Other types include backlogs like bugs and impediments that are directed to specific teams within the project.

Projects: Projects are containers that will help you manage the work. You can map your products to the project or a work project based on the scope of accomplishing your work.

Boards: Boards are provided for you to manage your work, each work displayed on the cards will have a set of columns dividing the tasks and status and when it will get completed.

Areas: Areas are a logical division of work to match the functional sections of the project. Each project is split into many sections like business, human resources, and risk management. This makes the project and boards more accurate and manageable for the different teams to work on them.

Notifications: Members of the organization need to be notified at times, therefore azure DevOps has provided a notifications feature to alert them.

Dashboards: Dashboards are what you will interact with when you log in to Azure DevOps, they contain work items and code. The dashboard offers test coverage and a wide range of widgets to show the iteration and progress of the projects.

Wiki: Wiki is a compilation of notes, requirements, and documentation, it reveals so much information that can help the practice of shared ownership and collaboration. They will convey useful information and serve as useful resource information for historical context with the team.

Analytics: The built-in analytics reporting feature will provide you with a detailed analysis of the software lifecycle using flow diagrams and charts. These data displays will show a broad perspective of your projects and all the narrow specific aspects.

Iterations: They are specifically used for planning and forecasting the project development, it may not present accurate results in the forecast. However, there are tools like SCRUM and KANBAN, which you will find very helpful in the planning.

So how does project management work?

Basically, you can set up one or more projects. Create teams. Add people (users) to teams. All users are split into groups and given certain permissions to update the code, change the deployment process, approve the build versions, and prepare to run tests.

Now your project should use a template for the processes and work items in order to become organized. You can arrange them in iterations or areas of one dimension. Areas are created for splitting the application project into logical works and iterations as a model for planning purposes. Obviously, these divisions are defined to work on something.

Once each task is completed, the developers will commit the code and trigger the build, which means the final code will be compiled and prepared for development. The deployment process can be performed manually or automatically in more than one environment.

The pipeline has the authority to run tests manually or automatically or decide to run manual tests before moving to the next stage. Pipelines can be totally configured to integrate any number of configurable steps on how and when the organization needs to function.

Maintain Source Code with Version Control.

Once a code is written by the programmer, it needs to be maintained safely by the managers from being corrupted or deleted. There are many instances where fixes in the code may not work, therefore the original code is kept along with the changed code to refer or start afresh.

Source control and version control can help to track and manage the code with all team members. The complete history of the code can also be viewed using source control, this will also help to resolve any conflict with developers as they try to change the name of the same file. The version control feature can help you to track the version of the software with certain bugs, as different versions can be labeled separately.

Types of version control management at Microsoft:

There are two kinds of version control management in Azure DevOps, a centralized version of control management is performed by using the tools called Team Foundation Version Control (TFVC) and a Distributed version control implemented with tools like GIT and GIT HUB.

Difference between TFVC and Git:

TFVC	Git
Code history on server	Code history on each team member's machine
Team member needs connection to server for check-in	Team member can commit code locally
Better for large code base	Better for relatively smaller code base but have a look at this link in case you have a big repo
Branches are heavier	Branches can be very light weight and only on local repo

Achieving Continuous Integration

Azure DevOps can achieve continuous integration and continuous development in your software project using the CI/CD pipeline. It facilitates the delivery of your software process in the Source, Build, Test, Merge and Deploy stages of the lifecycle.

Azure DevOps can automate the Ci/CD project to support the automatic build procedure which makes the code projects available to others immediately.

The CI/CD project pipeline stages are described in the illustration below:

Stages of a CI/CD Pipeline:

1. **Source Stage** – In most cases, when a change is attempted to the central repository, a pipeline run is triggered. These triggers are set by the CI/CD pipeline tool in the source stage.
2. **Build Stage** – The combination of source code and its dependencies when building into a runnable instance corporate to the end-user application. The built-in application languages like Java need compilation too, which is done in the build stage. If docker images are to be constructed, that can also be facilitated in this stage. Failing this stage marks a potential error in the code or its dependencies.
3. **Test Stage** – This stage corresponds to automated tests running to validate our code and its behavior accordingly. This stage acts as a sieve that prevents the bugs from reaching the end-user. There can be multiple stages, from smoke tests to end-to-end integration tests. Failure at this stage will expose errors in the code.
4. **Deploy Stage** – Once we have a runnable code, the deployment is processed with all predefined tests passed. There are a lot of stages like "Beta," "Staging," etc., for the product team. A "Production" stage for the end-users is also present.

Scheduling and Monitoring Team Progress in Azure DevOps:

Microsoft Azure DevOps has many options for monitoring the team's progress and schedule. Dashboards in the software suite support all features that will help customers and align with agile project management and methodologies. You can create any number of dashboards to manage your projects of work.

Dashboards are an improved work is an improved version of the team overview page, which can help provide real-time information's in an easy-to-read language. With one glance you can access totally informed decisions without the need for analysis or investigating other parts of the project.

Dashboards are the result of an agile software development process that will list out an information summary for the stakeholders and team. In addition, there are numerous widgets that you can use to make your work easier. They use the TFS Rest API to retrieve and display information on the project's work in progress. Widgets are highly interactive and easy to access information apps that will direct you straight to the work item.

Most widgets can be reconfigured to meet your needs and customized to your needs. The Delivery Plans extension developed by Microsoft labs will help the customer with his own software portfolio management, which can track multiple feature timelines, portfolio plans, and epic road maps.

