**What is Azure:**

* Microsoft Azure is a cloud computing service created by Microsoft for building, testing, deploying and managing applications and services through Microsoft-managed data centers.
* Software as a service (SaaS) platform from Microsoft
* Software as a service (SaaS) offering which was created to basically be a one-stop shop for implementing all your DevOps processes for our project
* Platform to implement all our DevOps Processes.
* It had many names before it became known as azure Devops, It was called team **foundation server then visual studio team services** so it had a bunch of other names before azure Devops. **Because it extended from existing services & tools** .
* **Devops Platform:** DevOps is many things. It is a combination of concepts and tools and basically anything that makes developing and releasing applications fast in an automated way and with high quality.
* Project need to implement Devops practices in order to achieve an efficient workflow.

**Services Azure Provided:**

* When you build an application and exposed on the internet people to use, so we need server, storage, facilities, database and full infrastructure.
* So instead of that we can get rental from cloud service provider.
* It will provide the products, Azure allow implements DevOps to practices, So we will get confidence of deployment pipeline and more..

**Azure Devops Tools:**

**AZURE BOARDS (PLAN) / AZURE REPOS (CODE) / AZURE PIPELINES (PACKAGE) / AZURE ARTIFACTS**

**Azure Boards**

* We can create tickets or tasks for features improvement or fixes for your applications as part of the agile or scrum processes. We can assign it to people to work on and we can also track progress of those features or improvements while they are being developed.
* Azure boards used for communicating between the developers, testers, product owner, etc..
* We can see all the status (Like, who is working on it, what stage it’s in, what’s its progress, etc..)
* Most popular workflows are agile and scrum

**Azure Repository**

* Azure repository feature is host your code in private Git repositories. Repositories like GitLab, GitHub, BitBucket, etc… which all are based on Git.
* Developers can host their code in azure repository and push their changes to it.
* Hosting code is one of the many features that they provide, so azure repository also its not just for hosting the code as part of the Devops life cycle developers collaborate and work together to develop high quality code.
* So in the repository as part of the Git workflow you have features such as pull requests, branches, various collaboration features and so on
* So when developer starts a task they create a temporary branch when done they create a pull request and other developers can review and comment on the pull request. They can communicate and collaborate until the pull request is good enough to be merged into the main branch.
* With azure repository you have the tool that enables you to implement whatever git workflow you choose to work with.
* Azure Repos commit, pull request or branch can be linked to the work items
* Once the features is developed and pull request is approved and merged into the main branch it need to be released so the end user can use it.
* So in order to release our code changes we first need to test it and package it into an artifact which is a deliverable that we can then deploy on the end environment.
* DevOps is all about automating things and workflows so that it’s fast and efficient so this process of testing and building the application is done by an automated CI or continuous integration process and building the CI pipeline azure has what’s called azure pipelines section.
* Pipelines can be written in YAML which means you can have your pipeline script as part of your code.
* The main building blocks of azure build pipelines are steps. For example if we want to test and package the application we may have steps to run tests , package application , build an image , push that image to an image repository so that later it can be deployed.
* So each step will execute a certain command to run the test, to package the application, build a docker image and so on.

**Pipeline Syntax: Step & Task**

**Step:**

* Main and smallest building block of a pipeline
* A step can either be a script or a task

**Task:**

* Task is a pre-created script offered as a convenience
* We can use the task with a set of inputs

**Pipeline : test🡪package🡪build image🡪push to repo**

**Job:**

* A job represent an execution boundary of a set of steps
* Each job runs on an agent
* All the steps in a job run on the same agent

**Agent:**

* Agent is a machine that will execute the tasks or the steps of the pipeline
* An agent is computing infrastructure with agent software installed on it.
* So overall the main task of the build pipeline is to test the code changes and if everything is fine produce an artifact that we can deploy

**Azure Artifacts**

* Artifacts differ based on the programming language used for writing the application, Could be a jar or war file for a java application / nuget file for .NET / ZIP fil TAR file etc..
* Here you can store and share different packages / From public and private sources
* Azure artifacts actually currently support three types of artifacts which are **maven packages, Nuget Packages and npm packages.** So if you are developing and building your application with any of these tools then you can store the artifacts produced in the build pipeline in the azure artifacts.
* In moden software development we use docker so no matter what language or tools used we always work with docker images
* If your build pipeline produces docker images you will connect your azure Devops to some docker registry like docker hub, azure container registry etc.. and basically store your images in that repository.

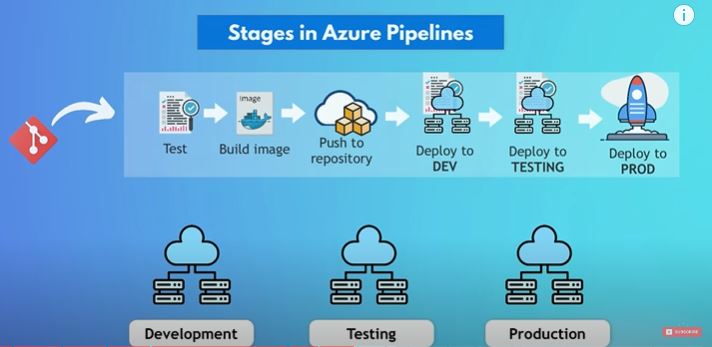
**Stages in Azure Pipeline**

**Stage:**

* A stage is a logical boundary in the pipeline
* Each stage contains 1 or more jobs
* By default, they run one after the other

**Deployment Jobs:**

* For deployment steps, it’s recommended to use the special deployment job
* Some benefits: Deployment History / Apply deployment strategy
* When we are deploying the new application version usually we don’t directly deploy to the production instead we deploy to intermediate environments we test it extensively and gradually promote it to the production when we’re almost 100% sure everything is fine.
* Common is to have development, testing and production environments.



**Azure Pipeline:**

Test🡪Build Image🡪Push to repository🡪Deploy to Dev🡪Deploy to Testing🡪Deploy to Production

**Azure Pipelines Templates**

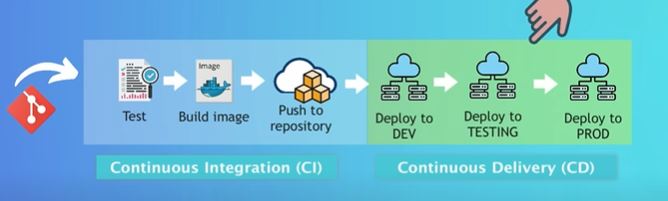
* You can create templates for jobs, stages or steps
* Template within templates are possible
* We want to write that logic once properly and then reuse it for all the applications that may need it
* In azure Devops pipelines YAML syntax we can actually put any code that is repeated and extract it in what’s called a **template** which is a separate file and can be referenced in the pipeline using template attribute and it can even be configured with parameters so it’s like a reusable piece of configuration that you can reference in different pipelines
* We can split our entire pipeline into multiple individual files and these files can be even be stored and managed in a dedicated separate repository.
* We can have a **template for a Job, a step or stage**

**Azure Pipelines Environments**

* Environment feature come in which is part of azure pipelines. we can create environments in azure Devops which will map to the actual deployment environments and then you configure in your pipeline which of these azure Devops environments you want to deploy
* Once the application gets deployed to the various environments you can actually view the deployment history per environment so this can actually be some additional valuable UI feature that give you a better overview of your deployments
* Deployment status can also be linked back to the original ticket so we have the additional information for the feature or improvement to which stages or environments it has been deployed to already.

**Release Pipelines**

* Deployment process or deployment part of the pipeline which is called CD or continuous deployment
* On azure Devops can also be built as a separate pipeline called release pipeline

****

**Classic UI or YAML**

* It’s usually a better idea to always have one CICD pipeline defined in YAML, instead of splitting that into two and have benefits of scripting our pipeline and making use of the reusable templates etc
* So the release pipelines is probably for more specific use cases. When we want to deploy existing artifacts from the artifact repository directly.
* We should have one pipeline for the complete CICD process.

**Azure Test Plans**

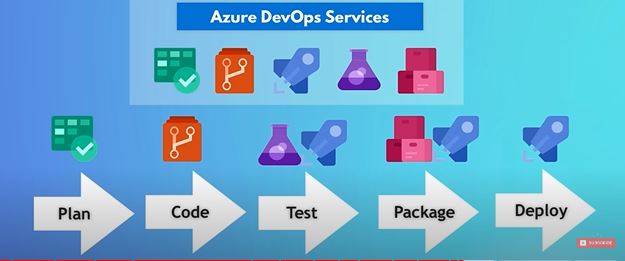
* Important part of an application release process is testing and you need to extensively test your code changes before deploying it to production
* For more complex application the more tests we need
* In azure Devops we have a dedicated section for tests
* A browser based test management tool
* Have a unified, central view of your test cases
* The main advantage of this is that we have own centralized place with an overview of all the test plans whenever releasing our application whether these are automated tests that were run in the pipeline or manual tests from developers, etc..
* We can even view and run the test cases from the Kanban board.
* When we are releasing our application these are automated tests that were run on the pipeline or manual test from developers, product owner testers, etc..
* We can see the results of those test executions in our feature descriptions to decide whether you can release the changes or not and we can view and run the test cases related to a feature directly from the **Kanban Board**.

**Azure DevOps Architecture**

* The pipelines executes tasks like **running tests🡪 Building Images🡪Deploy to development**
* We can execute these tasks on different machines with different operating systems

**Azure DevOps Services**

* Software as a Service (SaaS) from Microsoft
* Main part: Configurations, Pipelines, Repos created and stored
* The pipeline tasks themselves run on separate machine called agents, which are connected to the azure Devops services platform
* Azure offers to manage these agent machines by managed agents as well
* Microsoft manage the whole setup for us including the main service which holds the configuration , fast solution to executes the pipelines
* We can configure our own agents and connect them to the azure Devops platform
* Application development life cycle starting from planning the task all the way to developing and deploying it to the end environment



**Service Connections**

* In the DevOps process we often to connect and interact with other platforms
* For that, Azure DevOps needs to connect and authenticate to these platforms (like username and password or access token from these platforms that you need to make available in azure devops so that it can connect and authenticate itself with those platforms)
* In azure Devops we have called service connections feature which makes managing access to external platforms much easier.
* So less configuration efforts because we don’t have to create these credentials, Instead the credentials are created automatically when azure Devops connects to those services
* Another advantage is that it’s more secure because service connections use short-lived credentials, get generated on the fly when the connection is established
* Service connections can be created in the project settings section for administering the project. Here we can **manage settings** for all the features like **boards repositories, Pipelines, Test plans and Artifacts. /** We use self-hosted agents to run your pipelines and this is also where admins can configure these agents as well.

**Some Comparisions**

* Compare it with traditional CICD tools like Jenkins or modern ones like argo cd etc..
* The main difference here is that these are exclusively CICD tools

|  |  |
| --- | --- |
| Azure DevOps | Jenkins / Argo CD / AWS, etc |
| Covering whole DevOps Process including the CI/CD process. | For CI/CD process only |
| Less effort for integration, Because they are already integrated | We need to integrate these tool together, Like Build code repository, Jira Board, artifact repository etc,, |
| So we need effort in putting all these tools together like connecting git repository with Jenkins, connecting Jenkins with jira, to update status of feature tasks etc |
| Better traceability, Because we have linked data from all features. |  |
| Azure Devops is however gitlab because gitlab which started off as a git repository actually made a turn and decided to create an all-in-one Devops platform as well. Many of the features and use cases are pretty similar between gitlab and azure Devops | |
| AWS & Azure both are cloud platforms, Where we can create and configure your complete virtual infrastructure, create virtual server and use a bunch of other services as well | |
|  | * In AWS * More than 175 Services * Much More than DevOps |
| Azure platform and Azure DevOps platforms are more separated so we have 2 separate accounts for them / We can manage them separately and even use each platform without the other | All services in 1 place & 1 account |
| Integrate azure active directory in azure DevOps account / They’re two separate platforms but integrated with each other for various use cases |
| Azure pipelines works with any language, platform and cloud |  |
| Companies using azure, are more likely to use azure DevOps |  |

|  |  |
| --- | --- |
| **AWS DevOps** | **AZURE DevOps** |
| **SERVICE** | |
| Continuous delivery service for fast & reliable application updates | Services for teams to share code, track work, and ship software |
| **TECH-STACK CATEGORY** | |
| Continuous Deployment | Integrated Development Environment Tools |
| **FEATURES** | |
| Workflow Modeling | Agile Tools (Kanban Boards,etc..) |
| AWS Integration | Reporting (Dashborad, etc..) |
| Pre-Built Plugins | Git (Free private repo & pull req) |
| **USES** | |
| Simple Setup | Support Open Source |
| Managed Services | Integrations |
| GitHub Integration | GitHub Integration |
| Parallel Execution | Project Management Features |
| Automatic Deployment | Free For Stakeholders |
| Manual Setup Available | Jenkins Integration |
| **CLIENTS** | |
| PlayHQ | Microsoft |
| Affirm | Kingsmen Software |
| Quero Education | Evodeck Digital |
| Goopy | Mews |
| ReadyFor | Energy2Market |
| BitBank Inc | QRPoint |
| **INTEGRATION TOOLS** | |
| GitHub | GitHub |
| Jenkins | Git |
| Amazon EC2 | Docker |
| Amazon S3 | Slack |
| Aws Elastic Beanstalk | Jenkins |
| Run Scope | Trello |
| Cloud Bees | Visual Studio |