DevOps

Week 02

Murtaza Munawar Fazal

Azure DevOps

- Azure DevOps is a Software as a service (SaaS) platform from Microsoft that provides an end-to-end DevOps toolchain for developing and deploying software.
- It also integrates with the most-leading tools on the market and is an excellent option for orchestrating a DevOps toolchain.

What does Azure DevOps Provide?

- Azure DevOps includes a range of services covering the complete development life cycle.
  - Azure Boards: agile planning, work item tracking, visualization, and reporting tool.
  - Azure Pipelines: a language, platform, and cloud-agnostic CI/CD platform-supporting containers or Kubernetes.
  - Azure Repos: provides cloud-hosted private git repos.
  - Azure Artifacts: provides integrated package management with support for Maven, npm, Python, and NuGet package feeds from public or private sources.
  - Azure Test Plans: provides an integrated planned and exploratory testing solution.
- Also, you can use Azure DevOps to orchestrate third-party tools.

Github

- GitHub is a Software as a service (SaaS) platform from Microsoft that provides Git-based repositories and DevOps tooling for developing and deploying software.
- It has a wide range of integrations with other leading tools.

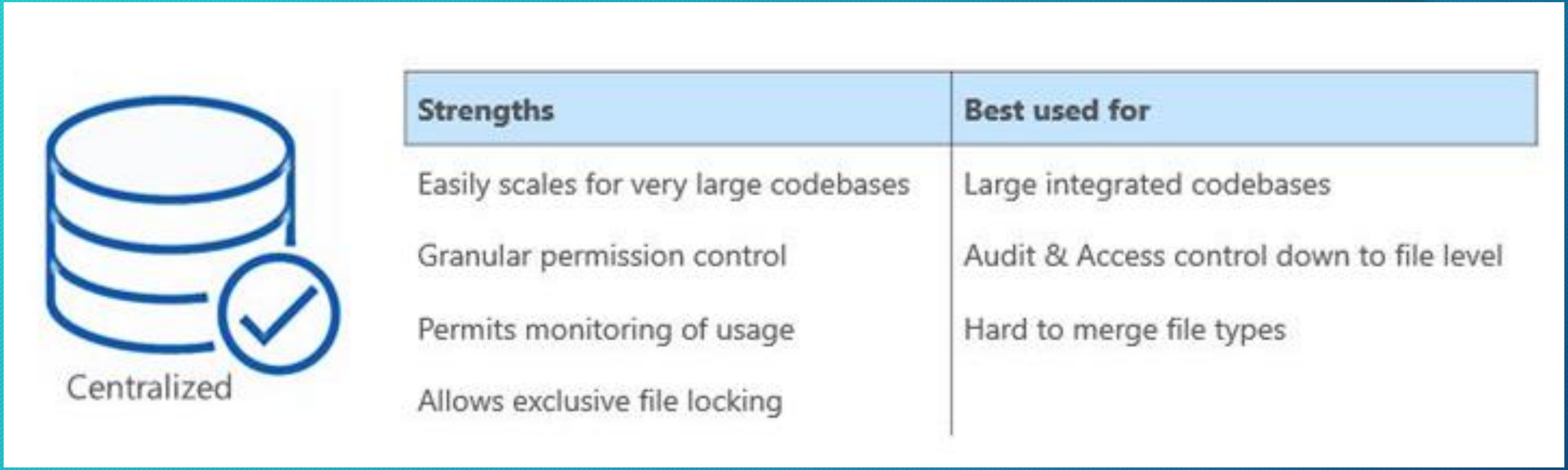
What does Github Provide?

- GitHub provides a range of services for software development and deployment.
  - **Codespaces**: Provides a cloud-hosted development environment (based on Visual Studio Code) that can be operated from within a browser or external tools. Eases cross-platform development.
  - **Repos**: Public and private repositories based upon industry-standard Git commands.
  - **Actions**: Allows for the creation of automation workflows. These workflows can include environment variables and customized scripts.
  - **Packages**: The majority of the world's open-source projects are already contained in GitHub repositories. GitHub makes it easy to integrate with this code and with other third-party offerings.
  - **Security**: Provides detailed code scanning and review features, including automated code review assignment.

Source Control

- A Source control system (or version control system) allows developers to collaborate on code and track changes. Use version control to save your work and coordinate code changes across your team. Source control is an essential tool for multi-developer projects.
- The version control system saves a snapshot of your files (history) so that you can review and even roll back to any version of your code with ease. Also, it helps to resolve conflicts when merging contributions from multiple sources.
- For most software teams, the source code is a repository of invaluable knowledge and understanding about the problem domain that the developers have collected and refined through careful effort.
- Source control protects source code from catastrophe and the casual degradation of human error and unintended consequences.

Centralized Source Control


Centralized

Strengths	Best used for
Easily scales for very large codebases	Large integrated codebases
Granular permission control	Audit & Access control down to file level
Permits monitoring of usage	Hard to merge file types
Allows exclusive file locking	

Centralized Source Control

- Centralized source control systems are based on the idea that there's a single "central" copy of your project somewhere. Programmers will check in (or commit) their changes to this central copy.
- "Committing" a change means to record the difference in the central system. Other programmers can then see this change.
- Also, it's possible to pull down the change. The version control tool will automatically update the contents of any files that were changed.
- Most modern version control systems deal with "changesets," which are a group of changes (possibly too many files) that should be treated as a cohesive whole.
- Programmers no longer must keep many copies of files on their hard drives manually. The version control tool can talk to the central copy and retrieve any version they need on the fly.
- Some of the most common-centralized version control systems you may have heard of or used are Team Foundation Version Control (TFVC), CVS, Subversion (or SVN), and Perforce.

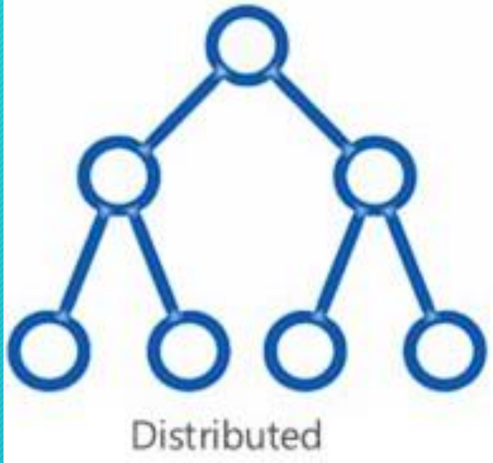
Centralized Source Control

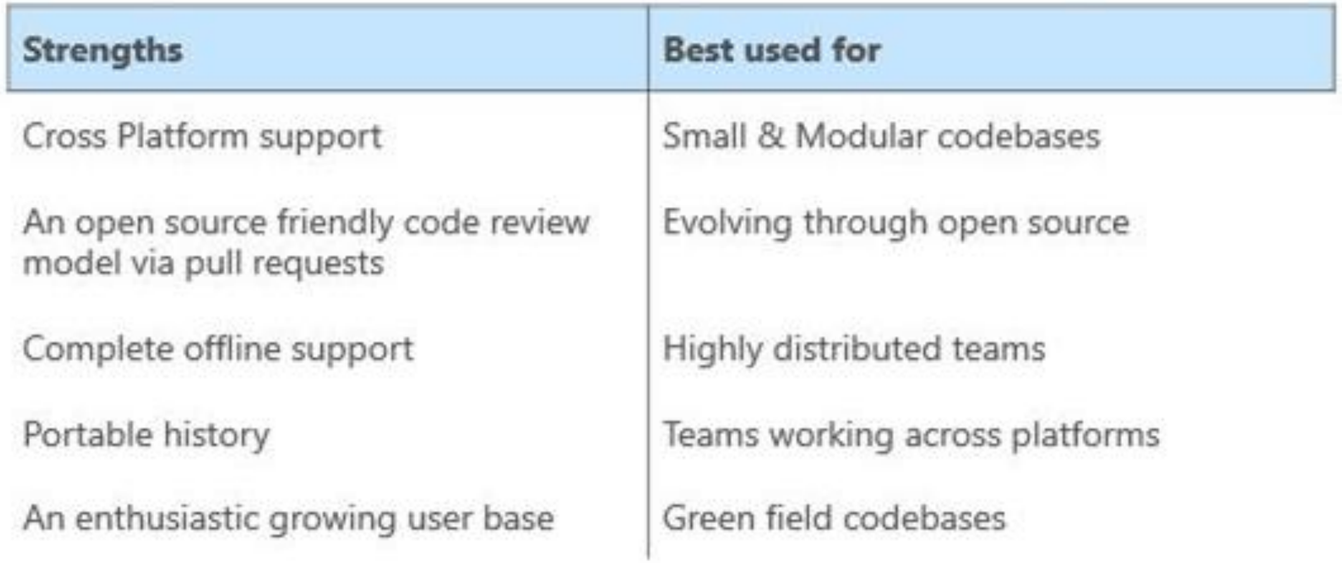
- If working with a centralized source control system, your workflow for adding a new feature or fixing a bug in your project will usually look something like this:
  - Get the latest changes other people have made from the central server.
  - Make your changes, and make sure they work correctly.
  - Check in your changes to the main server so that other programmers can see them.

Team Foundation Version Control (TFVC)

- Team Foundation Version Control (TFVC) is a centralized version control system.
- Typically, team members have only one version of each file on their dev machines. Historical data is maintained only on the server. Branches are path-based and created on the server.
- TFVC has two workflow models:
  - **Server workspaces** - Before making changes, team members publicly check out files. Most operations require developers to be connected to the server. This system helps lock workflows. Other software that works this way includes Visual Source Safe, Perforce, and CVS. You can scale up to huge codebases with millions of files per branch—also, large binary files with server workspaces.
  - **Local workspaces** - Each team member copies the latest codebase version with them and works offline as needed. Developers check in their changes and resolve conflicts as necessary. Another system that works this way is Subversion.

Distributed Source Control





Strengths	Best used for
Cross Platform support	Small & Modular codebases
An open source friendly code review model via pull requests	Evolving through open source
Complete offline support	Highly distributed teams
Portable history	Teams working across platforms
An enthusiastic growing user base	Green field codebases

Distributed Source Control

- These systems don't necessarily rely on a central server to store all the versions of a project's files. Instead, every developer "clones" a repository copy and has the project's complete history on their local storage. This copy (or "clone") has all the original metadata.
- The disk space is so cheap that storing many copies of a file doesn't create a noticeable dent in a local storage free space. Modern systems also compress the files to use even less space; for example, objects (and deltas) are stored compressed, and text files used in programming compress well (around 60% of original size, or 40% reduction in size from compression).
- Getting new changes from a repository is called "pulling." Moving your changes to a repository is called "pushing." You move changesets (changes to file groups as coherent wholes), not single-file diffs.

Distributed Source Control

- Doing actions other than pushing and pulling changesets is fast because the tool only needs to access the local storage, not a remote server.
- Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.
- Everything but pushing and pulling can be done without an internet connection. So, you can work on a plane, and you won't be forced to commit several bug fixes as one large changeset.
- Since each programmer has a full copy of the project repository, they can share changes with one, or two other people to get feedback before showing the changes to everyone.

Project Boards (Github)

- During the application or project lifecycle, it's crucial to plan and prioritize work. With Project boards, you can control specific feature work, roadmaps, release plans, etc.

- Project boards are made up of issues, pull requests, and notes categorized as cards you can drag and drop into your chosen columns. The cards contain relevant metadata for issues and pull requests, like labels, assignees, the status, and who opened it.

Types of Project Boards

- **User-owned project boards**: Can contain issues and pull requests from any personal repository.

- **Organization-wide project boards**: Can contain issues and pull requests from any repository that belongs to an organization.

- **Repository project boards**: Are scoped to issues and pull requests within a single repository.

Templates

- We can use templates to set up a new project board that will include columns and cards with tips. The templates can be automated and already configured.

Templates	Description
Basic kanban	Track your tasks with: To do, In progress, and Done columns.
Automated kanban	Cards automatically move between: To do, In progress, and Done columns.
Automated kanban with review	Cards automatically move between: To do, In progress, and Done columns, with extra triggers for pull request review status.
Bug triage	Triage and prioritize bugs with: To do, High priority, Low priority, and Closed columns.

Projects

- Projects are a customizable and flexible tool version of projects for planning and tracking work on GitHub.
- A project is a customizable spreadsheet in which you can configure the layout by filtering, sorting, grouping your issues and PRs, and adding custom fields to track metadata.
- You can use different views such as Board or spreadsheet/table.

GitHub Projects

- GitHub Projects allow you to control project deliverables, release dates, and iterations to plan upcoming work.
- You can create an iteration to:
  - Associate items with specific repeating blocks of time.
  - Set to any length of time.
  - Include breaks.
- It's possible to configure your project to group by iteration to visualize the balance of upcoming work.
- When you first create an iteration field, three iterations are automatically created. You can add other iterations if needed.

Project View

- Project views allow you to view specific aspects of your project. Each view is displayed on a separate tab in your project.
- For example, you can have:
  - A view that shows all items not yet started (filter on Status).
  - A view that shows the workload for each team (group by a custom Team field).
  - A view that shows the items with the earliest target ship date (sort by a date field).

Introduction of Azure Boards

- Azure Boards is a customizable tool to manage software projects supporting Agile, Scrum, and Kanban processes by default. Track work, issues, and code defects associated with your project. Also, you can create your custom process templates and use them to create a better and more customized experience for your company.
- You have multiple features and configurations to support your teams, such as calendar views, configurable dashboards, and integrated reporting.
- The Kanban board is one of several tools that allows you to add, update, and filter user stories, bugs, features, and epics.
- You can track your work using the default work item types such as user stories, bugs, features, and epics. It's possible to customize these types or create your own. Each work item provides a standard set of system fields and controls, including Discussion for adding and tracking comments, History, Links, and Attachments.