VS

by Max Yermakhanov

Max Yermakhanov

Senior Consultant

Microsoft Azure DevOps MVP

GitHub FastTrack Partner

Email: max@automagically.io

max@objectsharp.com

Blog: medium.com/@yermax

Twitter: @yermax

automagically

Before we start…

- Main question to ask is “Which tool is a better fit for my enterprise right now?”. Both tools are great, but very different
- We are comparing SaaS solutions only
- Software world is rapidly evolving
- Opinions are my own based on many years of hands-on experience
- It is important to remember that Microsoft is committed to supporting and investing in Azure DevOps while it executes its GitHub strategy

Overview

Azure DevOps

- --
  - Organization
    - **Team Project**
      - Repository (private, public)
      - Team

GitHub

- Enterprise Account
  - Organization
    - --
    - **Repository** (private, internal, public)
    - Team

Quick Decisions: Azure DevOps

- **Use Azure DevOps, if**:
  - You require to host data in the countries other than the United States (unless you are willing to host GitHub Enterprise Server on premises, of course)
  - You require integrated Test Management solution
  - You require to integrate with non-Git code repositories

Quick Decisions: GitHub

- **Use GitHub, if:**
  - You require GitHub Advanced Security or Dependabot
  - You require GitHub Codespaces
  - Your require mobile app
  - If you are starting from scratch

Azure Boards vs. GitHub issues

- Azure Boards provide robust planning capabilities to developers and non-developer user roles
- GitHub Issues (Projects and Discussions) provide developer focused project planning capabilities
- New GitHub Issues Beta feature has a lot of potential

Azure Repos vs. GitHub Repos

- Azure Repos code review experience is much cleaner, easier to navigate
- Azure Repos branch policies are more robust than GitHub branch protection rules. GraphQL for GitHub branch protection rules is excellent
- GitHub Repos are faster
- GitHub Repos offer dependency scanning and secret scanning
- GitHub Repos (thanks to GitHub Actions) allows you to build flexible and highly specific workflows
- GitHub Repos support Codespaces

Azure Pipelines vs. GitHub Actions

Scenarios:

- Continuous Integration (CI)
- Continuous Deployment (CD)
- Continuous Integration Continuous Deployment (CICD)

Azure Pipelines vs. GitHub Actions (CI)

- Both are excellent when it comes to CI
- GitHub Actions do not support templates
- No UI editor in GitHub Actions
- GitHub Advanced Security only available GitHub

Azure Pipelines vs. GitHub Actions (CD)

- Azure Pipelines provide mature features, I wish GitHub Actions had. For example:
  - Templates*, service connections, stages, gates, deployment groups, variable groups, secure files, etc.
  - Environments are not as mature as in Azure Pipelines and only available in GitHub Enterprise
  - No native way to download artifacts from other workflows
- Azure Pipelines allow to deploy from artifacts from Jenkins, TeamCity, CircleCI, and so on
- GitHub Actions is rapidly evolving and now offers feature I wish Azure Pipelines had. For example, secure deployments with OpenID Connect

Azure Pipelines vs. GitHub Actions (CICD)

- With Azure Pipelines you can have separate CI (build pipelines) and CD (release pipelines) workflows. In GitHub, both CI and CD workflows are defined in the same place
- There has been a lot said about CI and CD
- It’s worth noting that GitHub Actions are not just about CICD

Azure Artifacts vs. GitHub Packages

- Both GitHub Packages and Azure Artifacts offer artifact storage that will make you pay a small fee if you go over allowable limits
- GitHub Packages support containers (Docker and OCI images)
- Azure Artifacts support upstream sources
- Azure Artifacts support universal packages
- Azure Artifacts offer granular permissions control, better auditing and anonymous access

API, Integrations & Extensibility

- Both offer rich and well documented REST API. GitHub REST API is a bit more flexible and better documented.
- Azure DevOps has Microsoft Office integration
- GitHub has GitHub Apps to automate and improve workflows
- Both have myriad of integrations

Azure DevOps Licensing vs. GitHub Licensing

- Not a licensing expert. Licensing scares and bores me at the same time
- Azure DevOps is free for up to 5 basic users and unlimited number of stakeholders. Then, it's $6 per user/month
- Azure DevOps Basic license is included with a Visual Studio subscription
- GitHub pricing has layers (free for the basics, $4 per user/month for Team edition, and $21 per user/month for Enterprise Edition.) GitHub Advanced Security and GitHub Codespaces are extra
- GitHub Actions/Packages and Azure Pipelines/Artifacts might incur additional costs based on your consumption

Honorable mention

Azure DevOps

- Authentication and authorization
- az devops cli
- --
- --
- Azure DevOps Wiki
- Custom Dashboards and widgets
- Azure Test Plans
- --
- --

GitHub

- Authentication and authorization
- gh cli
- GitHub Advanced Security
- GitHub Codespaces
- GitHub Wiki
- GitHub insights
- --
- GitHub Pages

What Would I Choose?

- If I am starting from scratch, I would go all in on with GitHub
- If I need more advanced planning tool, I would integrate GitHub with Azure Boards
- If I need more mature release pipelines, I would integrate GitHub with Azure Pipelines
- If I am already heavily invested in Azure DevOps, but need features unique to GitHub, I would move my code to GitHub Repos
- If I am already heavily invested in Azure DevOps, and do not currently need any features unique to GitHub, I would stay put

Right Tool for the Right Job

Thank you

automagicolly