Microsoft Azure DevOps

Summary

Issue

We want to use devops to build, integrate, deploy, and host our projects. We are considering Microsoft Azure DevOps.

- We want developer experience to be fast and reliable, for the setup of the devops e.g. configuring as well as ongoing use e.g. fast build times.
- We want to consider using Microsoft Azure as whole, for hosting the project apps, databases, etc.

Decision

Decided against Microsoft Azure DevOps.

Status

Decided. Open to revisiting if/when new significant info arrives.

Details

Assumptions

All the usual devops assumptions, such as in the book Accelerate.

- Fast builds are a significant help. This accelerates the feedback loops.
- We can swap in/out pieces from alternate vendors i.e. we may want to bring our our higher-speed build servers, or use our own choice of version control system, or coordinate with a self-hosted continuous integration server.
- Streamlined usabilitiy is a signficant help, for developer experience, and in turn for subtle areas such as consistency, clarity, security, and ease of learning curve.
- When anything is broken or problematic, we want an effective way to report the issue. This is especially important for any security-related issues.

Constraints

None known. Azure has a published commitment to playing well with external tools.

Positions

We considered using Microsoft Azure Devops vs. AWS which is incumbent.

We experimented with Azure DevOps, Azure Pipelines, Azure Repo, and Azure spin up of new server via Terraform.

We experimented with getting support from Microsoft representatives.

We gathered information from peers on blogs and Hacker News.

Argument

Azure DevOps advertises an excellent set of offerings, but they do not hold up, and they do not work well together, and support is poor.

Our firsthand experience:

- Azure setup is a mess of UIs, some of which overlap with Microsoft accounts, some of which don't. E.g. there's an Azure sign in, a Microsoft.com sign in, a Live.com sign in, etc. and all are simultaneously in play.
- We encountered a minor security issue during setup, and found no resolution. We tried many ways to report it, to many Microsoft reps, with no success. We successfully reported it to Microsoft security, which replied with won't fix.
- Documentation is often either wrong or outdated. At least some of this is due to Microsoft's poor search engine, and some of this is due to sub-par SEO.
- Terraform setup is well documented, and works. However, Terraform supoprt is weak compared to AWS because Microsoft is building business relationships with vendors to do chain-through Terraform setup examples.

Our peer experiences:

- After we did our own blind assessment, we looked for peer experiences. What we found confirmed our experiences.
- Peers reported additional problems with build times, and problems with bring-your-own-build server. These problems are signficantly more severe than UI problems, because doing builds is the core purpose of a build pipeline, and we expect to do many per day.
- We found excellent participation by Azure teammates in the discussion areas. Kudos to Microsoft for this. We are especially impressed with Edward Thomson, Azure PM and coder, because of his participation, directness, and technical explanations.

Implications

Choosing Microsoft Azure DevOps looks likely to be more expensive (~3x) in time and cost than not choosing Azure.

Related

Related decisions

If we choose Azure DevOps, there are many related offerings, including Azure Repo, Azure Pipeline, etc. We believe that if we choose Azure Devops, this may make it easier to use more Azure capabilities, or may make it harder to use other vendors' capabilities.

We believe that Microsoft is making great strides in developer experience, and we see Microsoft making large acquisition of developer tools (e.g. GitHub) and dependencies (e.g. Citus).

If we choose Azure DevOps, then we may want to emphasize choosing the Microsoft acquistion offerings, and we may also want to approach the acquisition offerings with more care/assessement because of potential tissue-rejection e.g. staff turnover risk.

Related requirements

We want build times to be very fast. We accept paying a high premium for this. This is because we want to iterate very fast.

We want reliability to be very high. We accept paying a high premium for this. This is because we are testing high-value use cases, including financial transations, confidential transactions, etc.

Our top 4 devops KPIs include mean time to recovery, which necessitates fast builds and high reliability.

Related artifacts

We want the build system to output artifacts suitable for using in other systems, such as Artifactory.

Related principles

Easily reversible. We can evaluate Azure DevOps in parallel with AWS incumbent.

Notes

Microsoft Devops CI: An Unsatisfying Adventure

https://toxicbakery.github.io/vsts-devops/microsoft-devops-ci/

Blog post.

"As a software developer, I know first-hand how difficult it is to build quality products quickly and cheaply. It’s an art form that we sometimes get right, and other times devolves into something akin to the Obama era healthcare government site. Our level of control over the resulting product varies, and blame for failure often falls on the wrong people in the decision-making hierarchy. Microsoft’s Azure DevOps (formerly known as Visual Studio Team Services), despite clearly good intentions, is a perfect storm of bad decisions and poor execution."

Hacker News discussion highlights

https://news.ycombinator.com/item?id=18983586

"We use Azure DevOps extensively at my work and, after having used GitHub, Gitlab, self hosted solutions, Jenkins, TeamCity... Azure DevOps ranks dead last."

"The UI is terribly clunky everywhere. The worst for me are pull requests. Incredibly tough to work with people on a pull request. I can't even point you to "a" particular problem - for us it's broken everywhere."

"Azure Devops is something I want to love. The UI keeps changing, but doesn't fix underlying bugs that have been around for ages."

"The tools are not well integrated, the UI is really slow, there’s no dashboard view of active pull requests, builds, releases, etc for my favorite repos. Build/Deploy times are insanely slow."

"We tried to also use Azure Boards (Work Items, Boards, Backlogs, etc). Ouch. It is a complete UI mess of disjointed ideas. Instead of implementing one thing well, they implemented two dozen things terribly."

Windows Development MVP

Windows Development MVP here. I feel like I must shoulder some of the responsibility here for not being louder about these issues. But must say, I'm disappointed to hear you're "surprised" about the UX issues. I've been telling your folks the UX is dreadful (e.g. as far back as pre-launch) and kept hearing back "we know, we're fixing it". I'll start formalizing the feedback and push it through the pipes, stay tuned. I'm also local (Bellevue), would love to come in and try to pipeline our relatively simple oss .net/wpf/uwp app. I suspect it'll be an eye opener for the both of us.

Some examples:

- You can't build a pipeline with a git repo. that contains submodules
- Found it impossible to edit the PATH for some custom tooling
- The New Pipeline experience just doesn't make a lot of sense, new users clicking around will eventually end up at the wrong Docs.

Edward Thomson (Azure PM) summary

I wrote the code that merges your pull requests. Program Manager for at Microsoft for Azure DevOps; formerly a software engineer on version control tools at GitHub, Microsoft, SourceGear.

https://www.edwardthomson.com/

Co-maintainer of libgit2. https://libgit2.github.io

Co-host of All Things Git, the Podcast about Git. https://www.allthingsgit.com/

Curator of Developer Tools Weekly, a newsletter about development tools. https://developertoolsweekly.com/