DevOps Fundamentals

.NET

DevOps is the union of people, process, and products to enable continuous delivery of value to end users.

HTTPS://DOCS.MICROSOFT.COM/EN-US/AZURE/DEVOPS/LEARN/WHAT-IS-DEVOPS

What is DevOps?

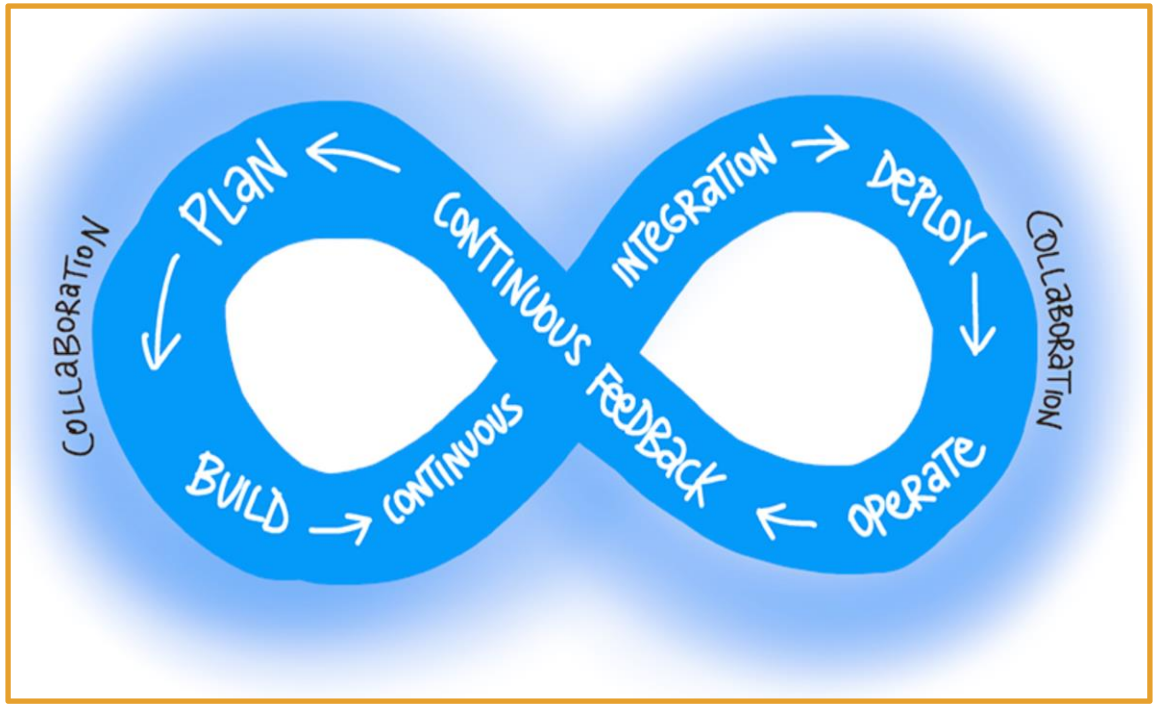
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

The contraction of “Dev” and “Ops” refers to replacing “Siloed” DEVelopment and OPerations Teams.

With DevOps, multidisciplinary teams work together with shared, more efficient practices and tools.

Essential DevOps practices include:

- Agile planning,
- Continuous Integration,
- Continuous Delivery, and
- monitoring of applications.



Who is DevOps?

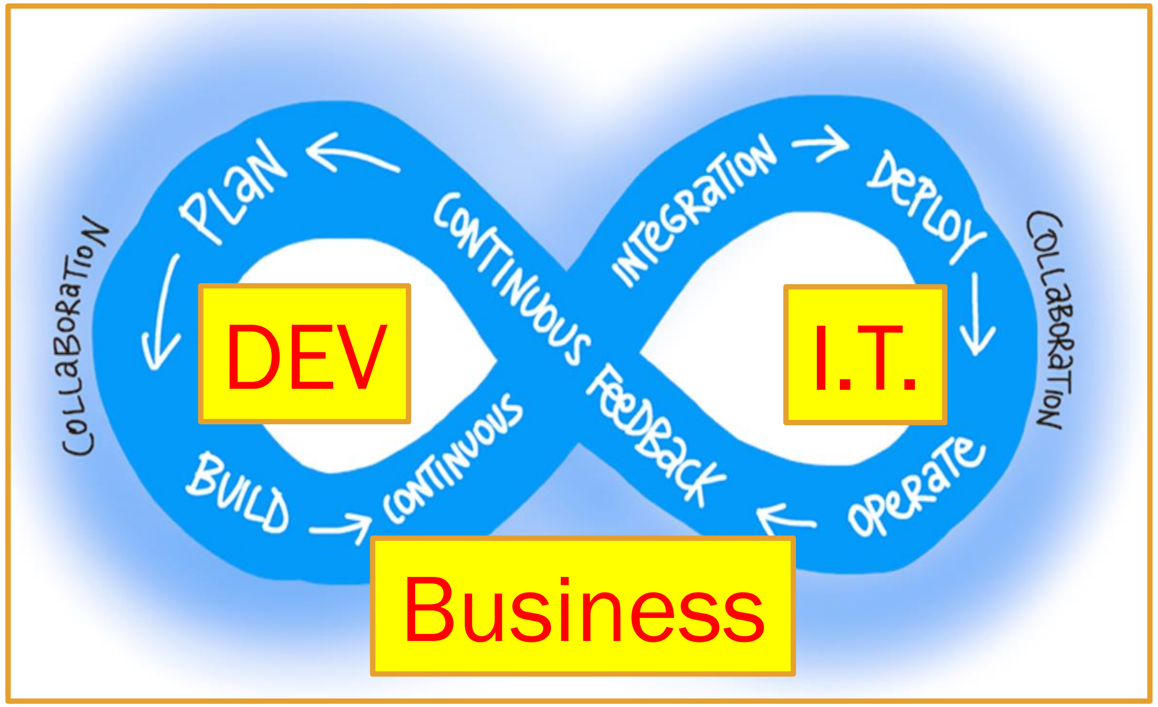
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

DevOps is the combination of the processes of the:

- Business team,
- IT team,
- Development team

In ***DevOps***, these teams form a feedback loop that has a shared goal.

- The Dev team plans and builds the app.
- The IT team deploys and maintains the app.
- The Business Team verifies that the correct product is created and delivered.



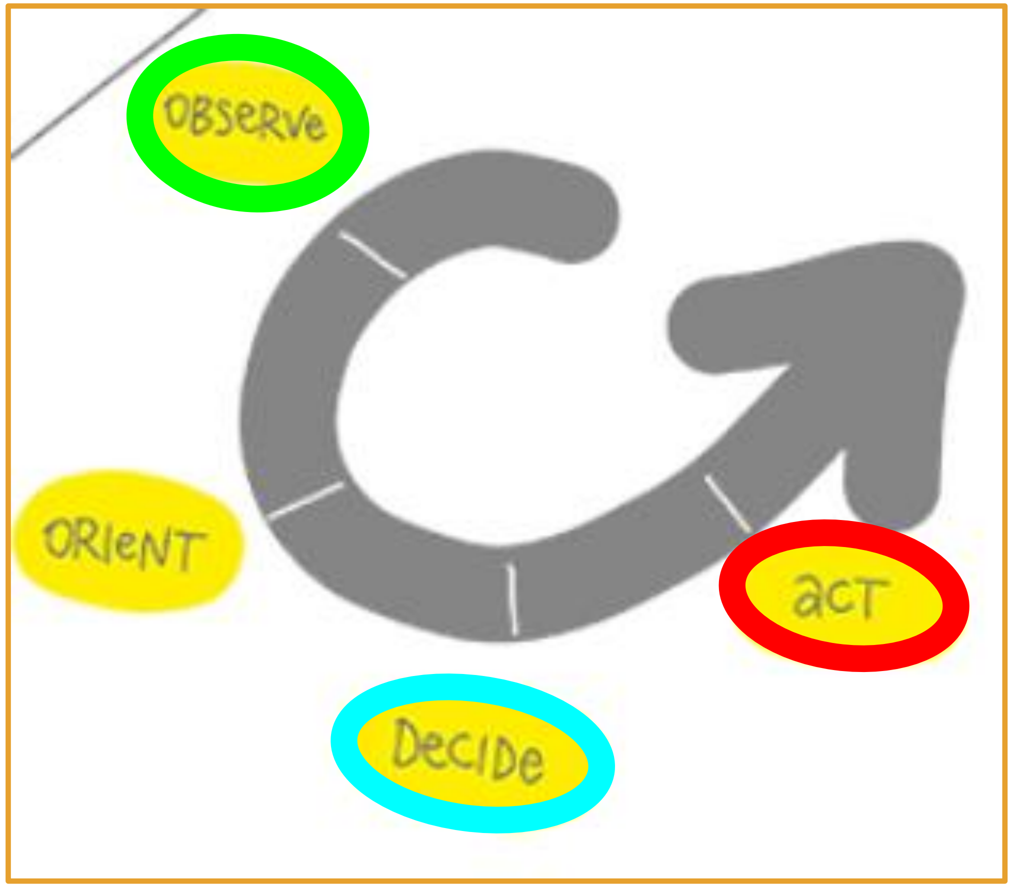
DevOps and The O.O.D.A. Loop

https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops#understand-your-cycle-time
http://www.slideshare.net/adriancockcroft/speeding-up-31799721

The OODA loop:

1. O - observe business and market needs and current user behavior.
2. O - orient with the options for what you can deliver.
3. D - decide what goals to pursue.
4. A - act by delivering working software to real users.

The four OODA Loop steps occur in a ***Cycle Time***. The Cycle repeats until a project is complete.

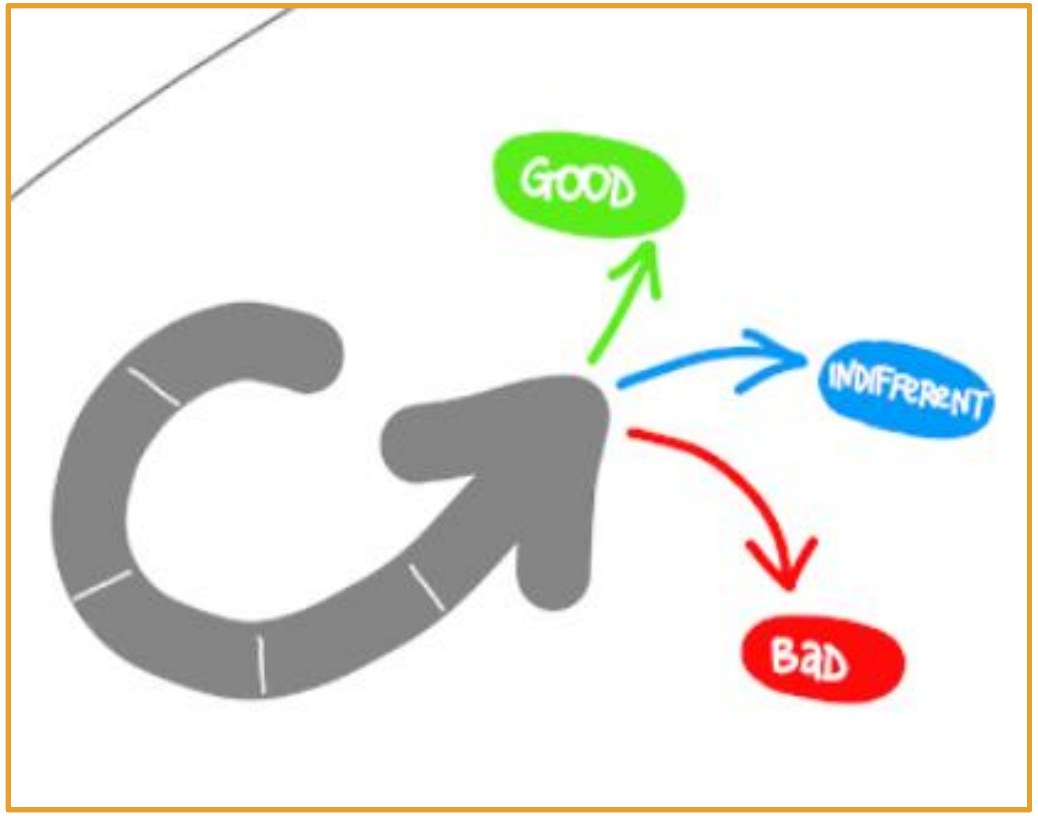


The OODA Loop - Cycle Time

https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops#understand-your-cycle-time
http://www.slideshare.net/adriancockcroft/speeding-up-31799721

Your ***Cycle Time*** is determined by how quickly you can complete the four steps.

The ***feedback*** that you gather with each cycle should be real, actionable data. Something should be learned from each cycle. This is called ***Validated Learning***.



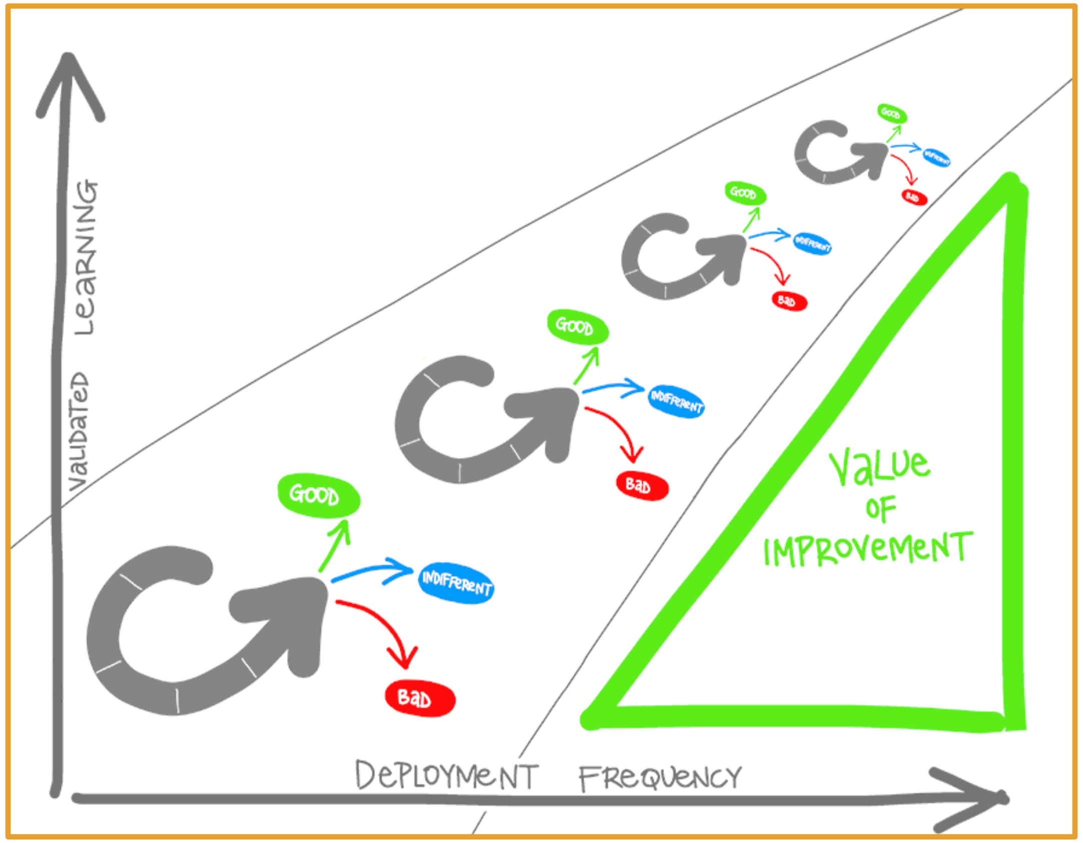
DevOps shortens Cycle Time

https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops#shorten-your-cycle-time

When ***DevOps*** practices are adopted, smaller, more focused teams will:

- use more automation,
- improve the release pipeline, and
- deploy more frequently.

The more frequent the deployment, the more experimentation can be done, and the more opportunity there is to gain ***Validated Learning*** after each cycle.

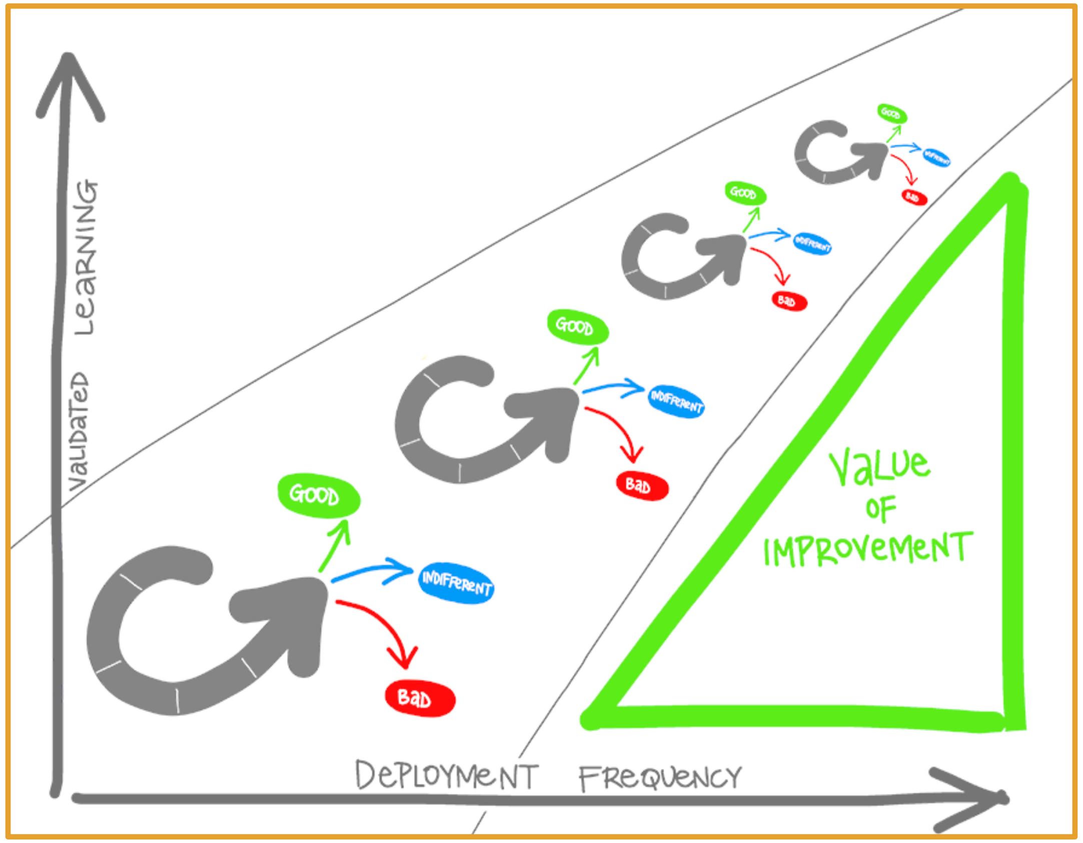


Achieving Devops

https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops#how-to-achieve-devops

The overall goal is to shorten the project ***Cycle Time*** to zero.

This is achieved through ***Continuous Integration and Continuous Delivery (CI/CD)***.



CI- Continuous Integration

https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops#how-to-achieve-devops

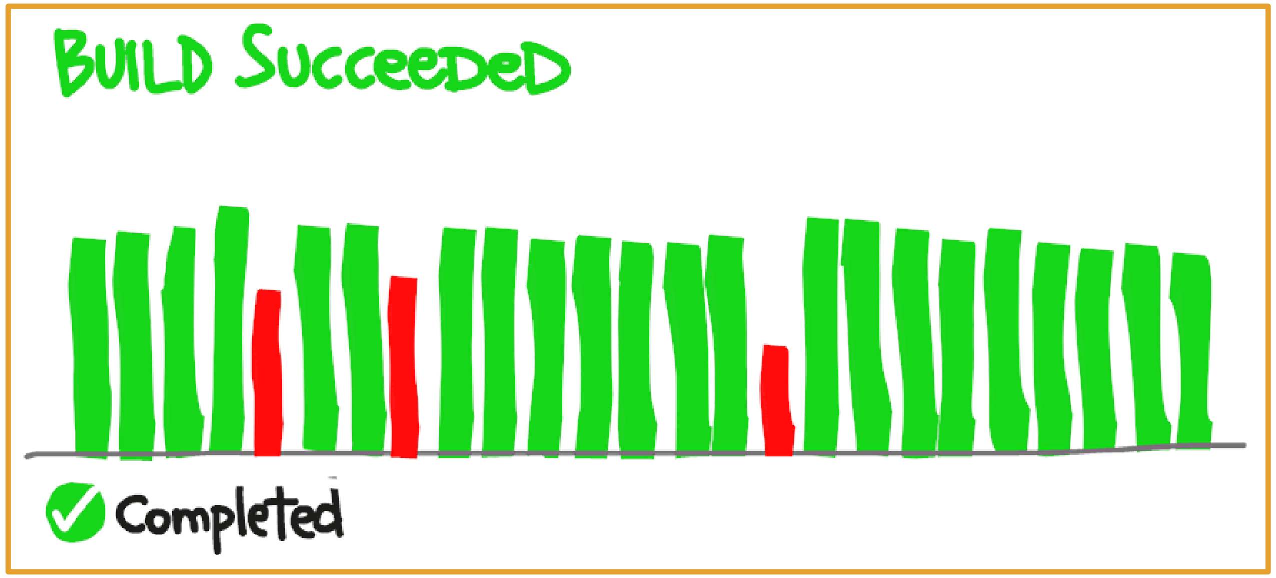
Continuous Integration (CI) is the process of automating the build and testing of code every time a team member commits changes to version control (GitHub). Ideally, changes are committed multiple times per day. Developers merge even small changes to version control.

To achieve *Continuous Integration*, the commit of new code triggers an automated build system to grab the new code from the shared repository and build, test, and validate the full master branch.

Constantly merging code avoids

- merge conflicts,
- duplicated efforts, and
- divergent strategies.

A developer submits a "pull request" when a feature or change is complete. The changes are accepted and merged into the master branch. Then the feature branch is deleted.



CD – Continuous Delivery

https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-delivery

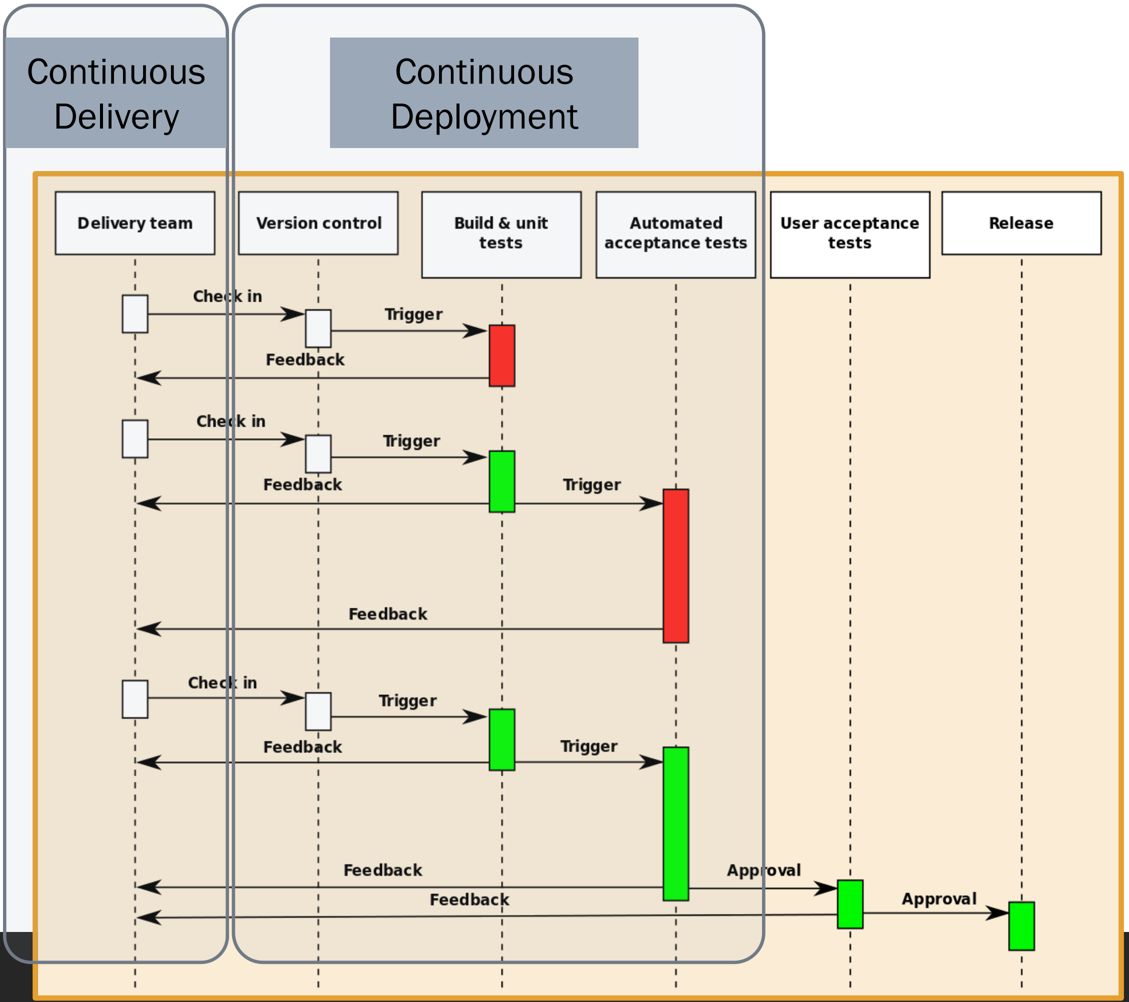
Continuous Delivery (CD) has been shown to achieve the shortest path from new code to final deployment.

CD is the process of building, testing, configuring, and deploying code to a production environment.

A ***Release Pipeline*** is made up of multiple build, test, or staging environments which are used to automate the deployment. Automation is preferred because manual processes are unreliable and produce delays and errors.

Without ***Continuous Delivery***, software release cycles become a bottleneck for dev teams.

An automated ***Release Pipeline*** allows a "fail fast" approach to validation, where tests fail quickly so code can be immediately refactored.

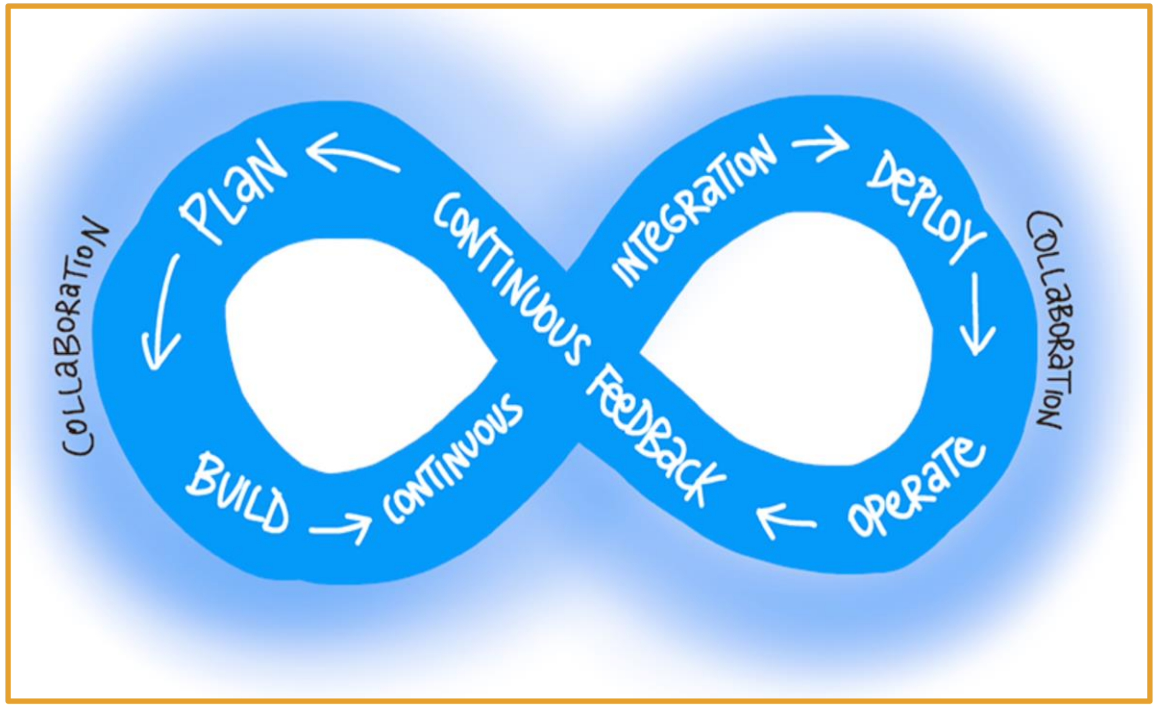


Continuous Delivery vs Continuous Deployment

https://aws.amazon.com/devops/continuous-delivery/

Continuous Delivery is when code changes are automatically prepared for a release to production. ***Continuous Delivery*** expands upon ***Continuous Integration*** by deploying all code changes to a testing environment and/or a production environment after the build stage.

When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process and I ready for manual testing.

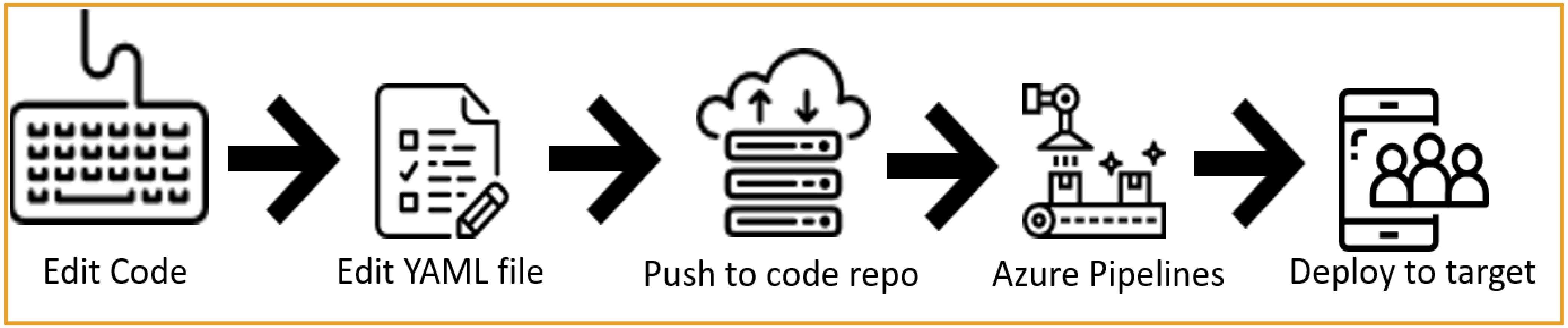


Azure DevOps - Introduction

https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/dotnet-core?view=azure-devops
https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops
https://docs.microsoft.com/en-us/azure/devops/pipelines/?view=azure-devops

Azure Pipelines is a cloud service that you can use to automatically build and test your code and make it available to other users. *Azure Pipelines* works with many language or project types.

Azure Pipelines combines *Continuous Integration (CI)* and *Continuous Delivery (CD)* to constantly test and build your code to be shipped to any target.

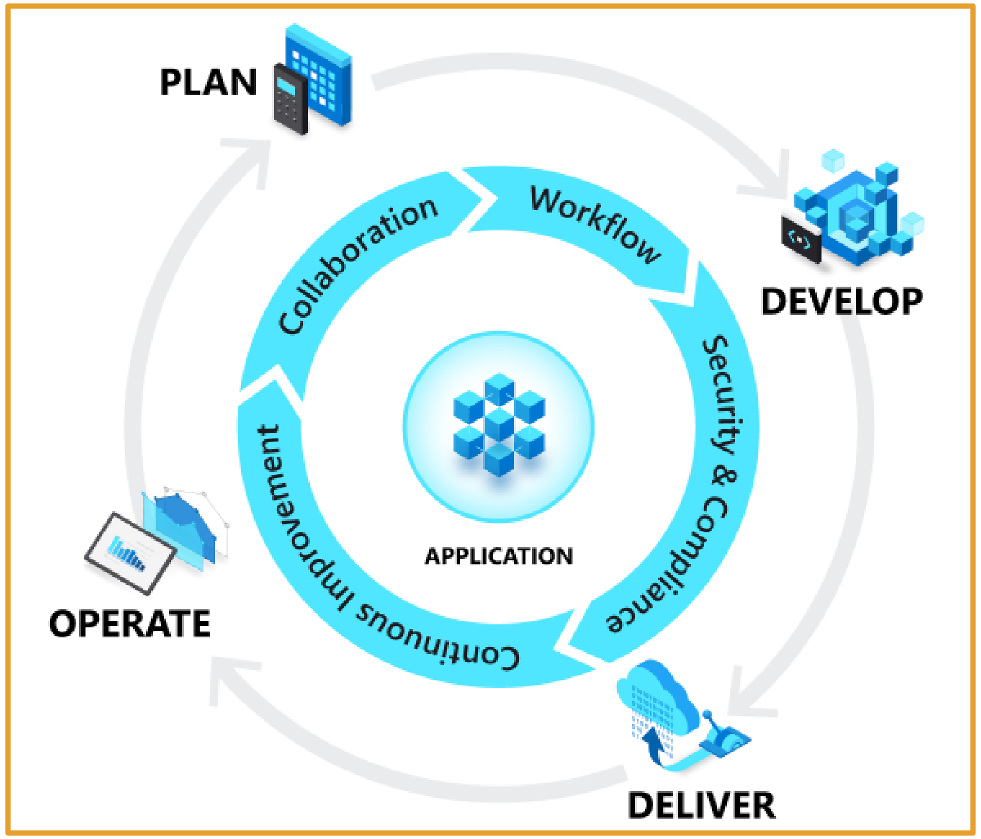


Build Definition

https://docs.microsoft.com/en-us/aspnet/web-forms/overview/deployment/configuring-team-foundation-server-for-web-deployment/creating-a-build-definition-that-supports-deployment#task-overview

A ***build definition*** is the mechanism that controls how and when builds occur. ***Azure DevOps*** uses a .yml file to define a build. Each build definition specifies:

- The things you want to build.
- The criteria that determine when a build should take place
- The location to which the Build should send build outputs.
- The amount of time that each build should be retained.
- Various other parameters of the build process.

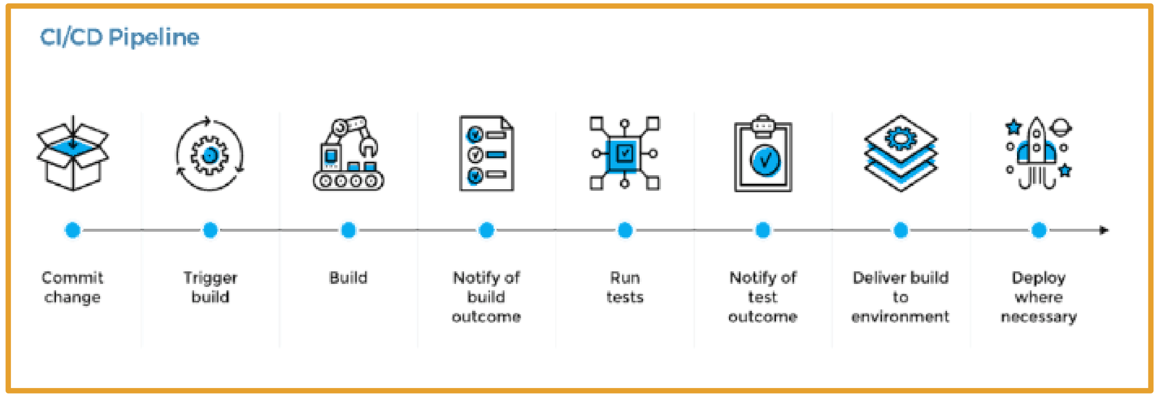


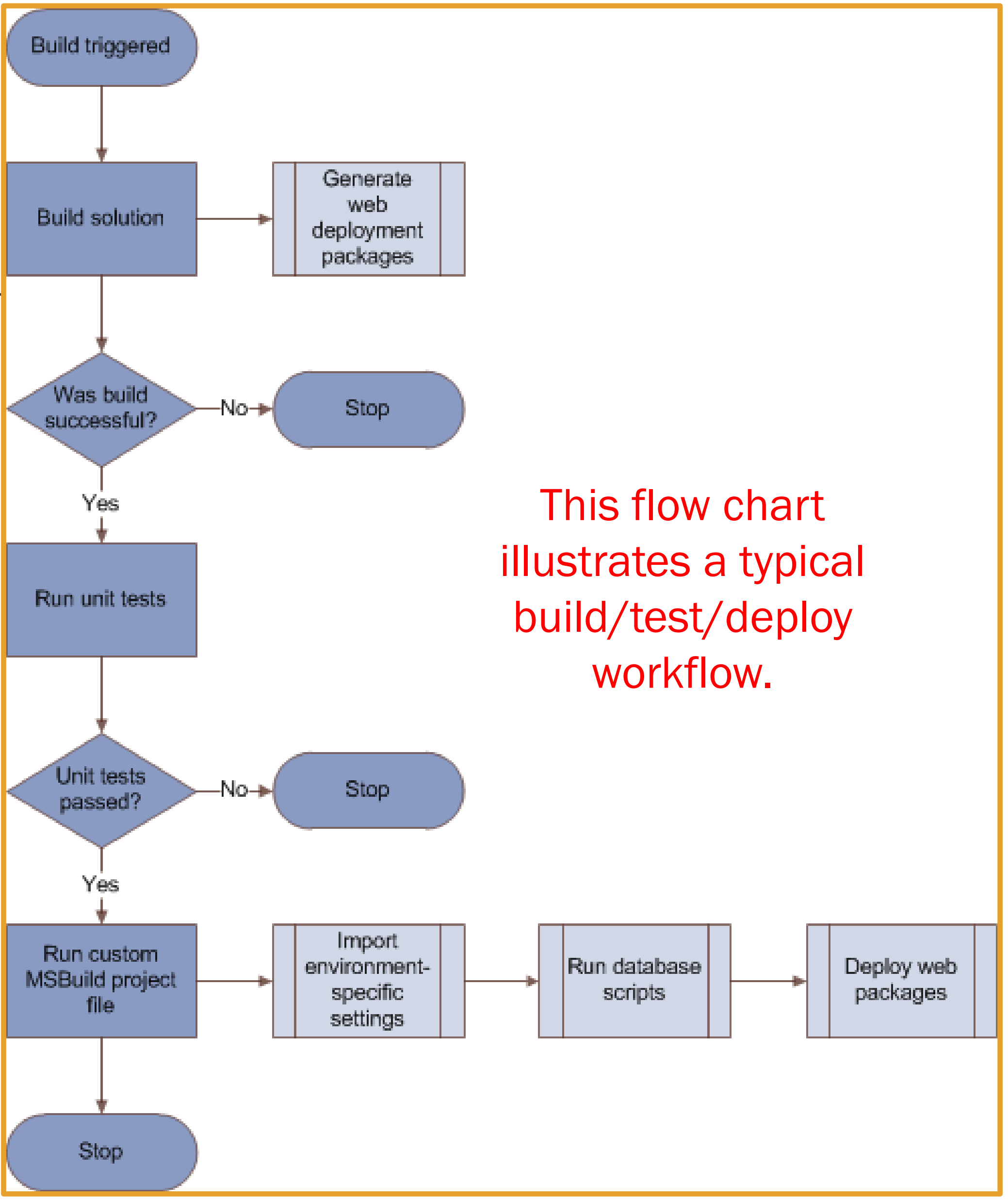
Release Pipeline

https://docs.microsoft.com/en-us/azure/devops/pipelines/release/?view=azure-devops

Release pipelines in ***Azure Pipelines*** help your team implement CI/CD and deliver software to your clients faster and with lower risk.

You can fully automate the testing, delivery, and analysis of your software all the way to production or set up semi-automated processes with required approvals and on-demand deployments.





This flow chart illustrates a typical build/test/deploy workflow.

Pipeline Monitoring and Logging

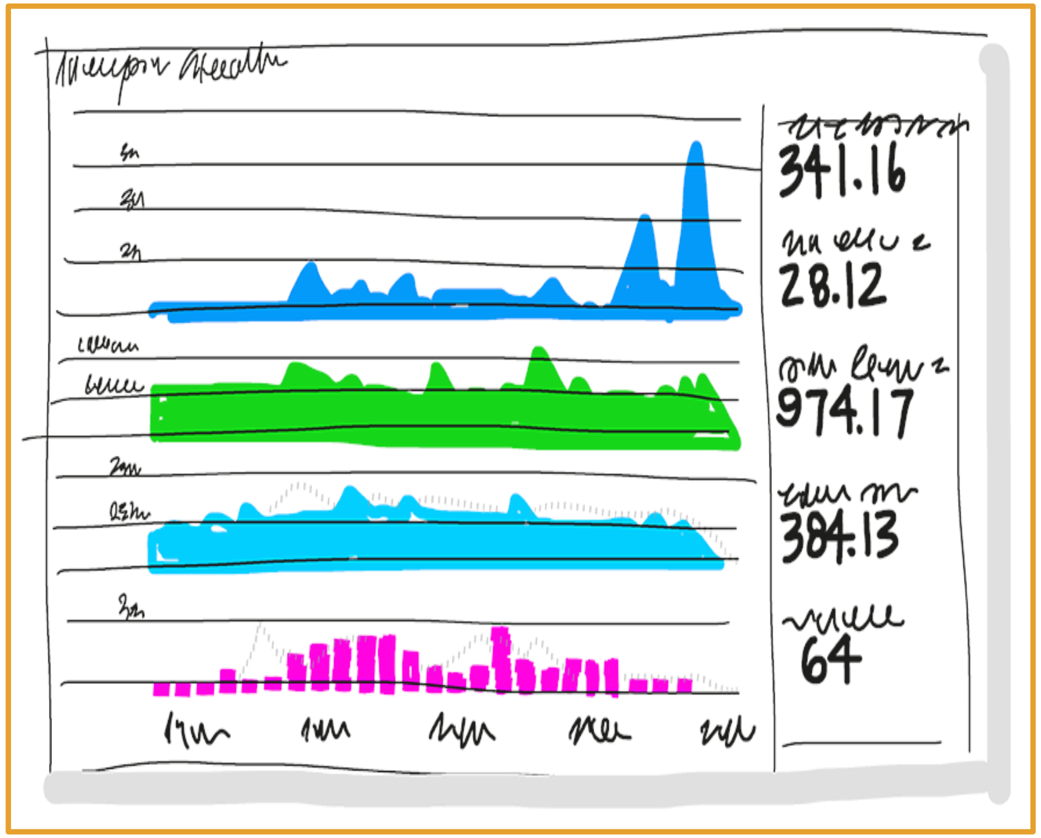
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-monitoring

Monitoring should be built into the Pipeline to allow “test in production”.

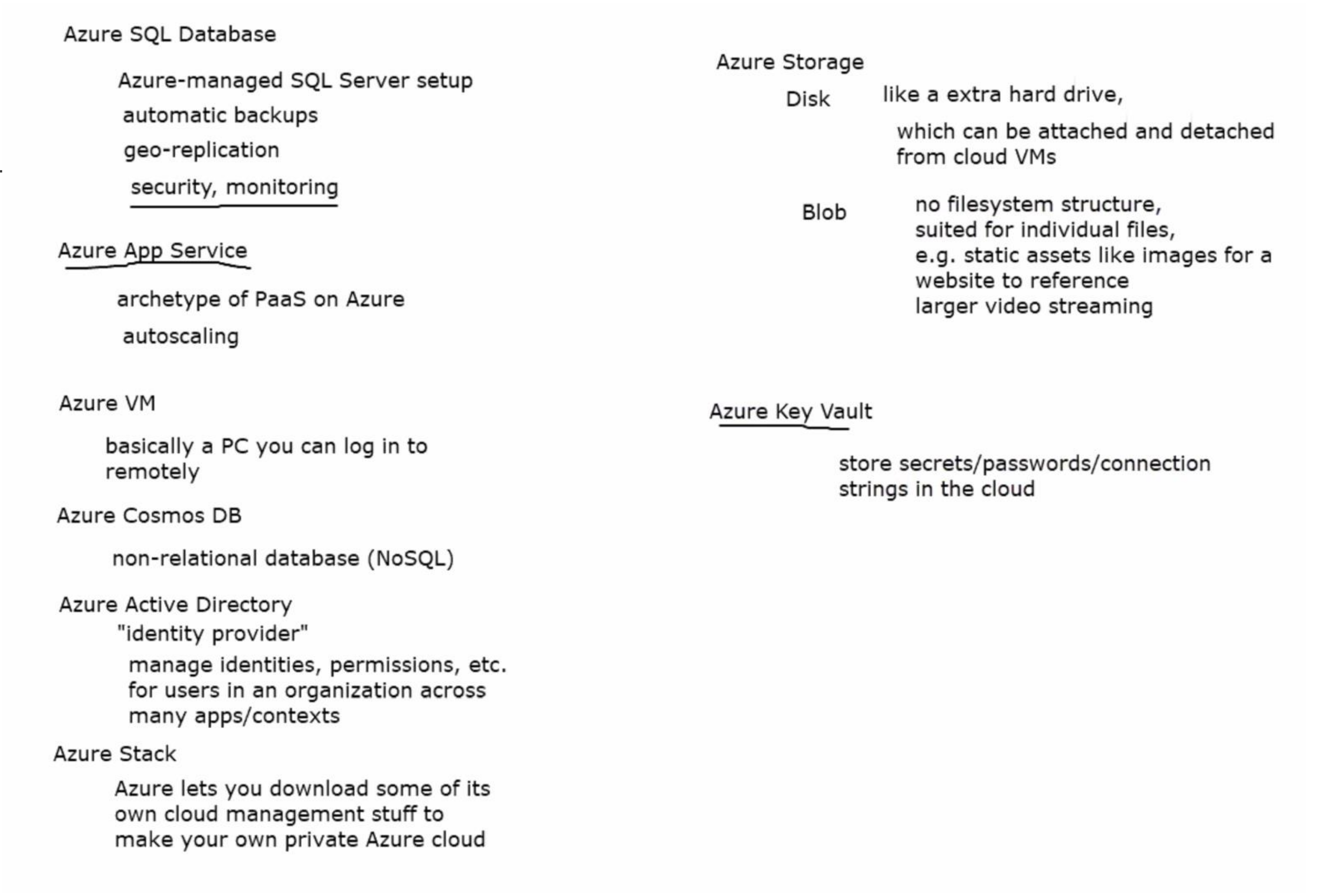
Monitoring enables ***Validated Learning*** by immediately delivering details about an application’s performance and usage patterns.

Issues are immediately fed back to development teams via the automated build, test, and report phases in the process. The team can quickly pivot their strategy if needed.

There are various third-party sites to which the pipeline can report testing code coverage and code quality analysis.



Azure SQL


Azure SQL Database

Azure-managed SQL Server setup
automatic backups
geo-replication
security, monitoring

Azure App Service

archetype of PaaS on Azure
autoscaling

Azure VM

basically a PC you can log in to
remotely

Azure Cosmos DB

non-relational database (NoSQL)

Azure Active Directory
"identity provider"
manage identities, permissions, etc.
for users in an organization across
many apps/contexts

Azure Stack

Azure lets you download some of its
own cloud management stuff to
make your own private Azure cloud

Azure Storage

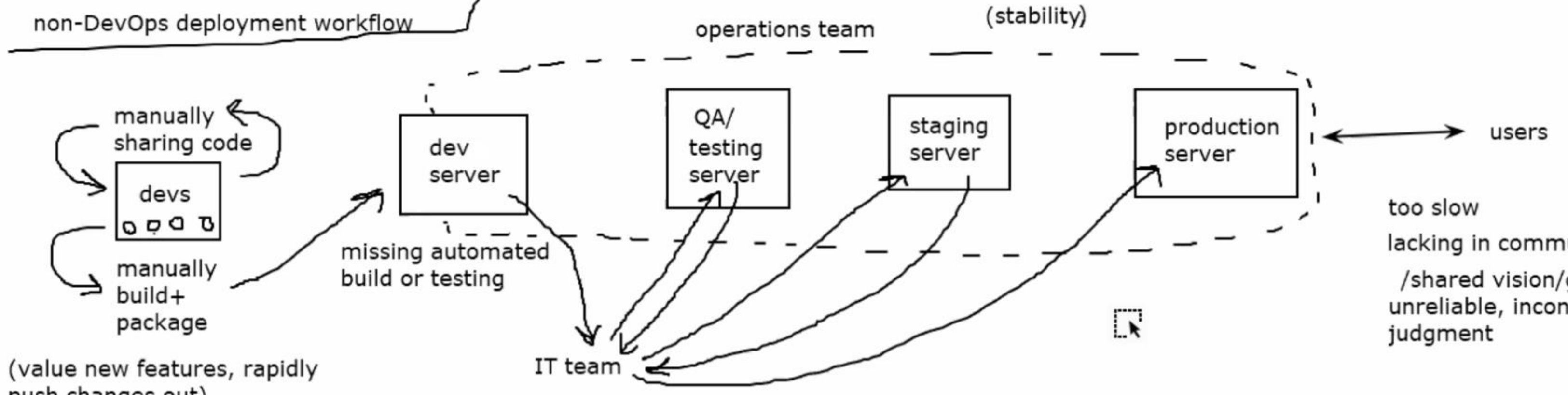
Disk like a extra hard drive,
which can be attached and detached
from cloud VMs

Blob no filesystem structure,
suited for individual files,
e.g. static assets like images for a
website to reference
larger video streaming

Azure Key Vault

store secrets/passwords/connection
strings in the cloud

non-DevOps deployment workflow



(value new features, rapidly push changes out)

too slow
lacking in communication
/shared vision/goals
unreliable, inconsistent human judgment

rapid cycle time

bring all stakeholders into the conversation

bring devs and ops together to agree on processes to automate the integration and delivery of software

minimum basic level of DevOps

continuous integration (CI)

very often (multiple times a day at least)
newly written code is integrated with all the other devs code
(in a source-control repository like git for example)

automated buliding and testing of the latest code in the repo

analyzing that code, anything to automatically ensure its quality just from being pushed to master branch

usually achieved these days with CI tools e.g. Jenkins, CircleCI, AWS CodeBuild, Azure Pipelines

deploy to a dev server (automatically)

"level 2"

continuous delivery ("CD", "CDe")

we have CI, *and*,
the deployment is automated all the way through to the production server except for manual checks along the way.

"level 3"

continuous deployment ("CD")

like continuous delivery, except no manual approval needed

DevOps