DevOps Fundamentals

.NET CORE

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?

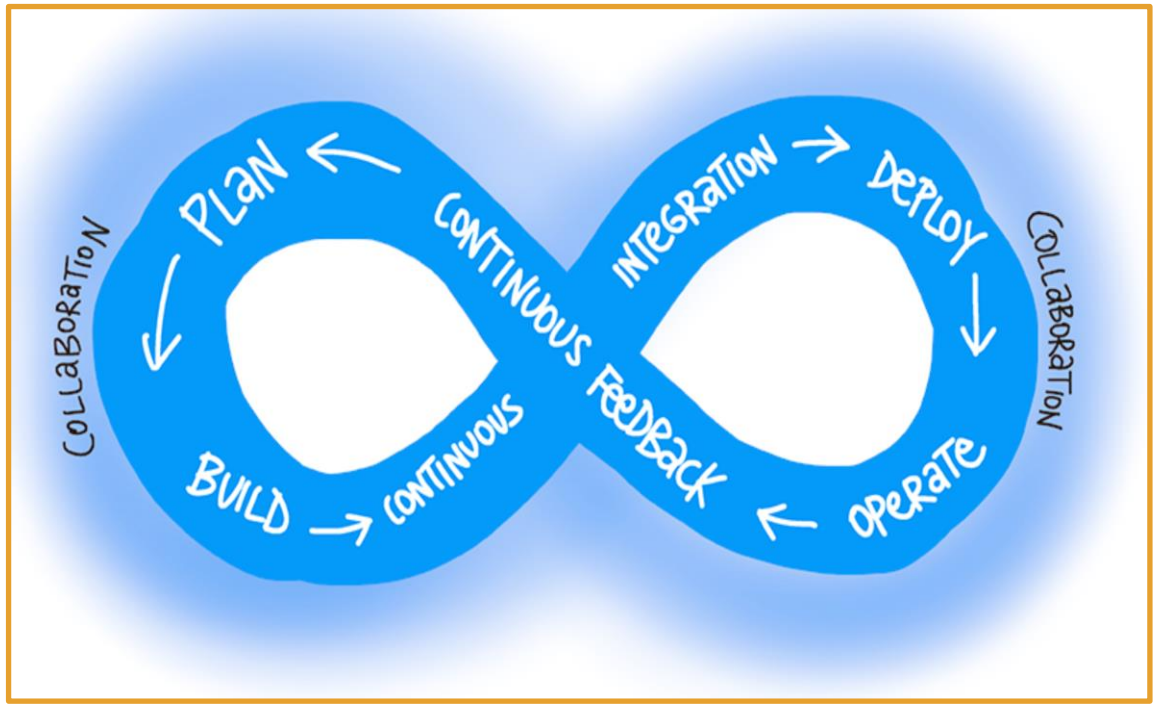
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

The contraction of “Dev” and “Ops” refers to replacing “Siloed” DEV-elopment and OP-erations Teams.

Instead, multidisciplinary teams that work together with shared, more efficient practices and tools are created.

Essential DevOps practices include:

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



Who is DevOps?

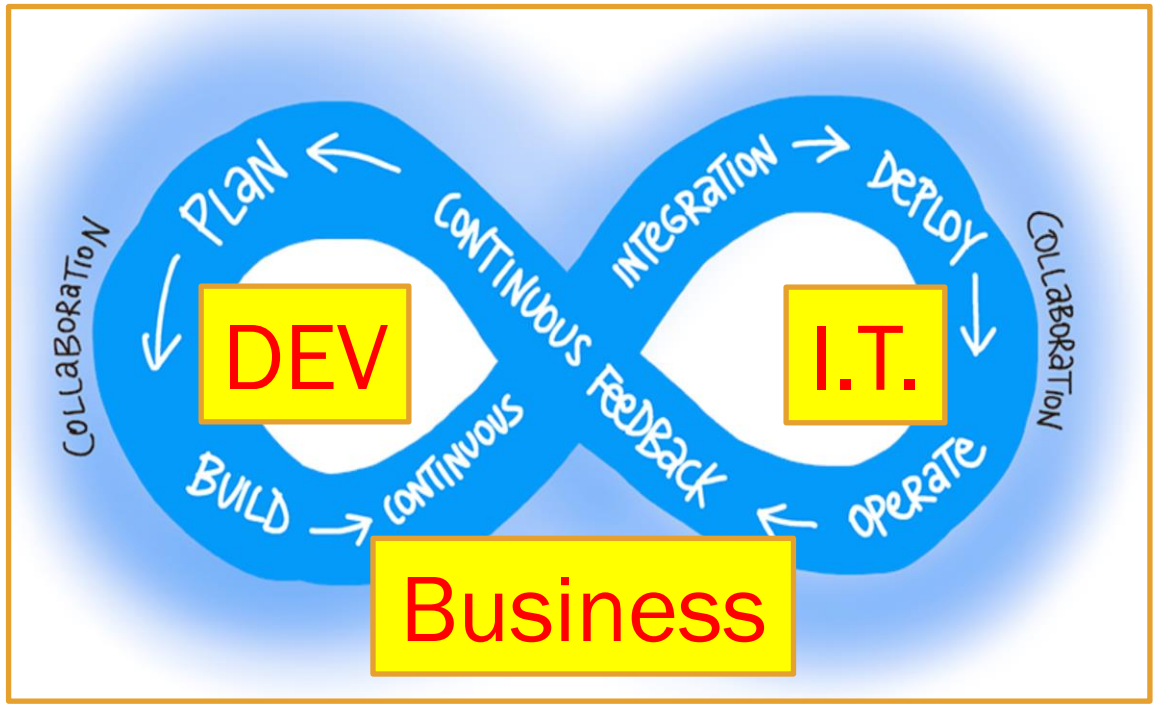
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

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 singular 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.



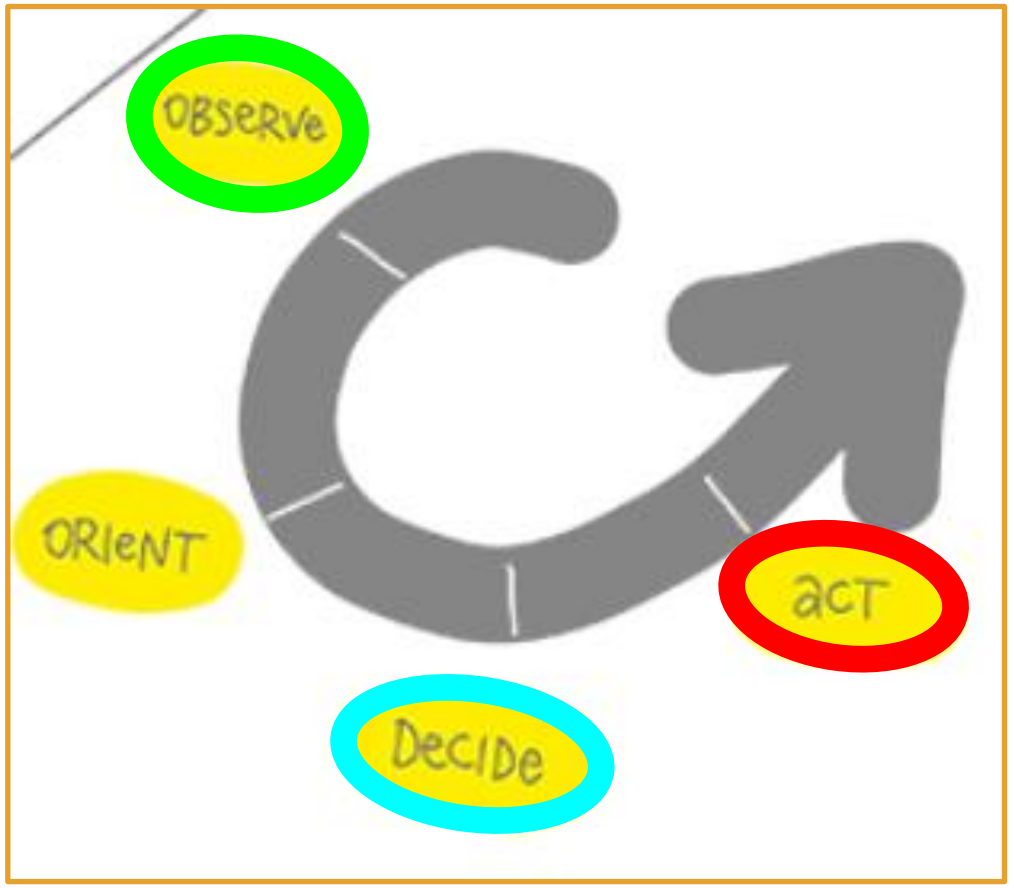
DevOps and The O.O.D.A. Loop

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.

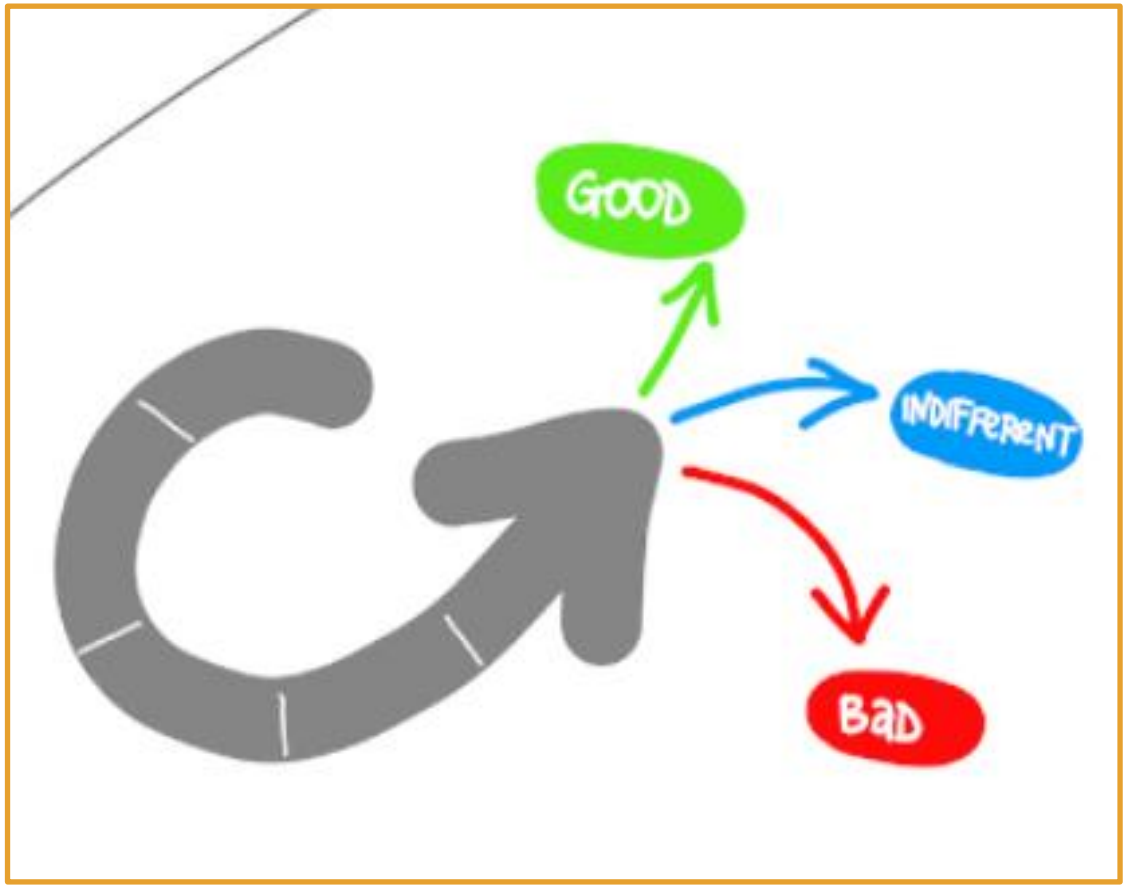


The OODA Loop - Cycle Time

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***.



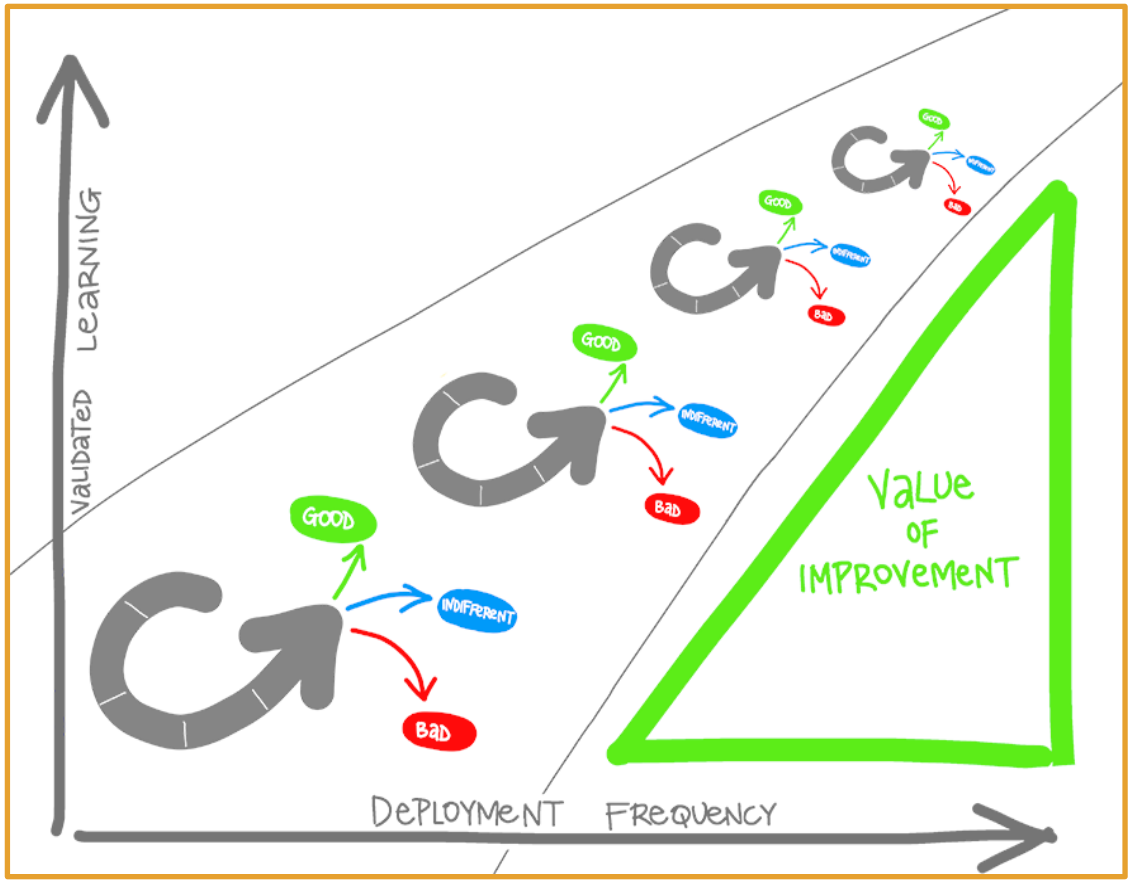
DevOps shortens Cycle Time

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.

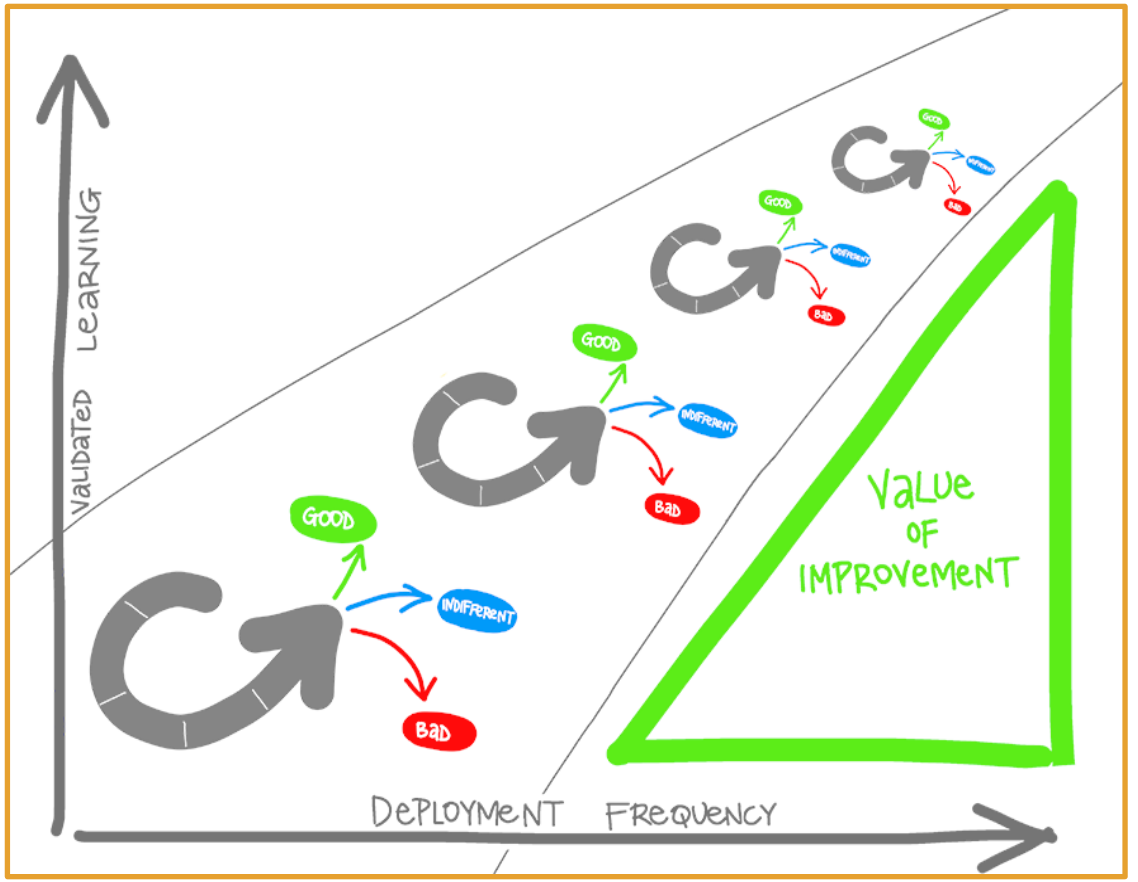


Achieving Devops

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

The overall goal is to eventually 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. 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
- divergant 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

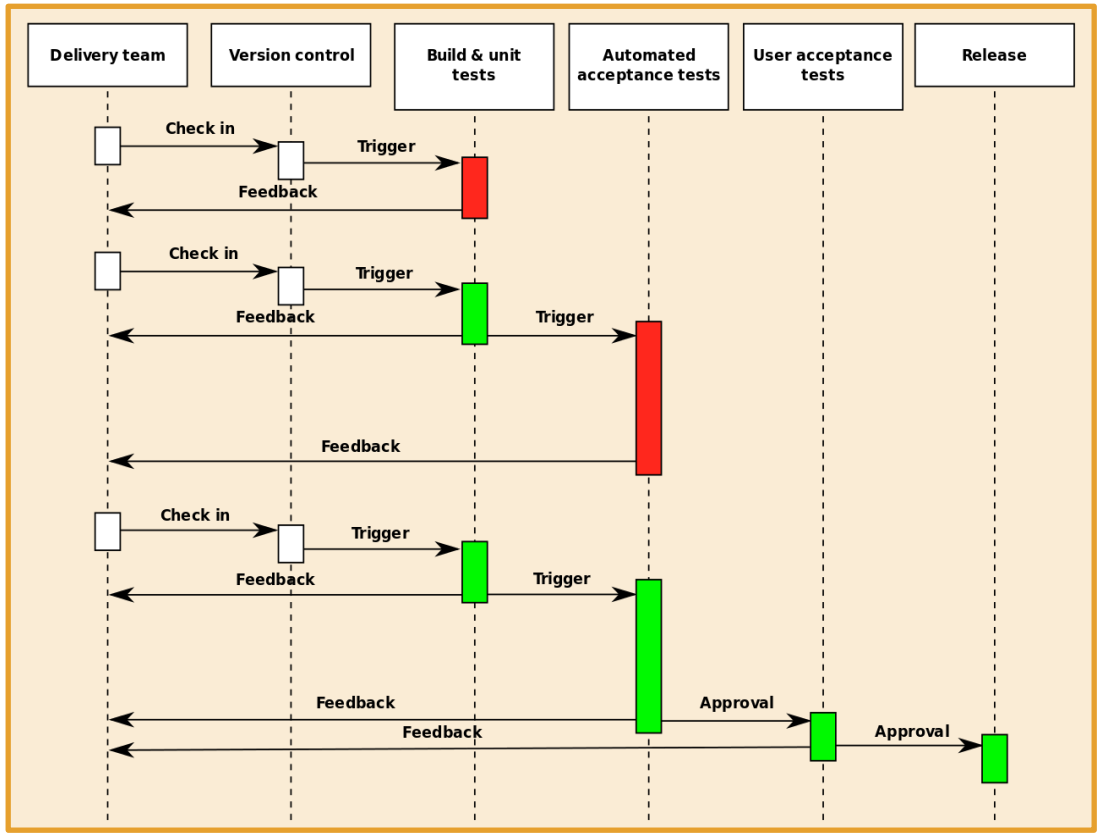
Continuous Delivery (CD) has been shown to achieve the shortest path from new code to final deployment.

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.



Pipeline Monitoring and Logging

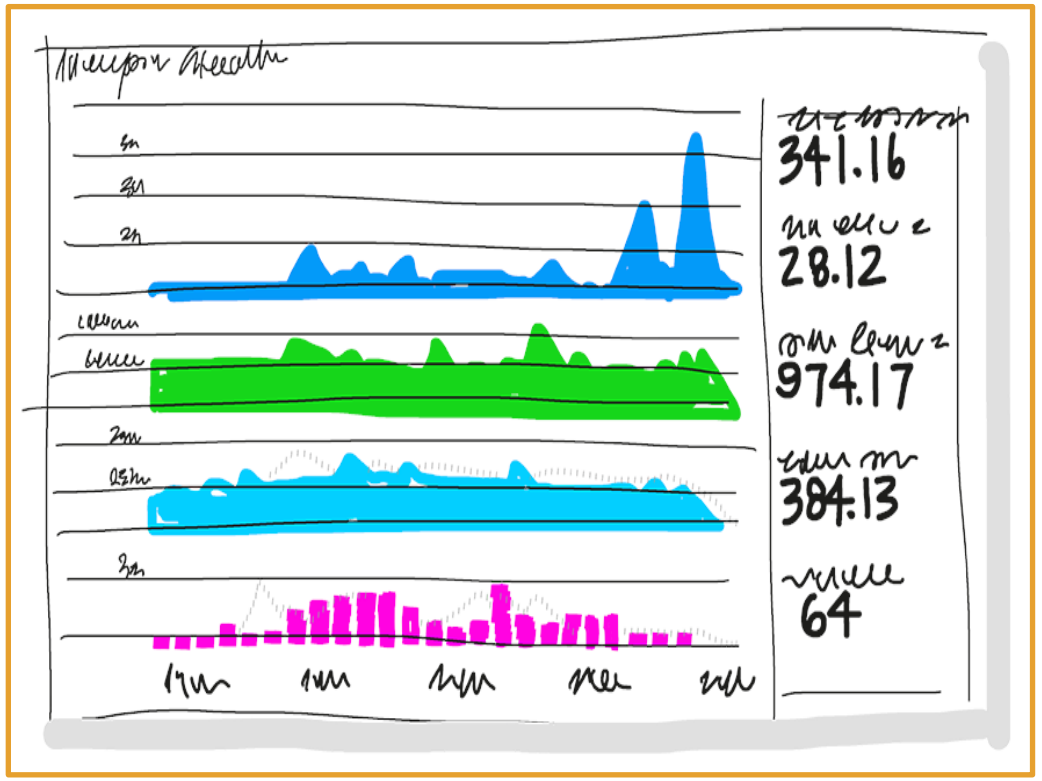
https://docs.microsoft.com/en-us/azure/devops/learn/what-is-monitoring

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.



End of presentation

What is the pipeline?

https://www.gocd.org/getting-started/part-1/

Show them the “This is a pipeline” image.. To break down the process of what a pipeline does. Here they are called tasks… in Azure they are called “Stages” inside the stages there are “Jobs”.

https://docs.gocd.org/current/introduction/concepts_in_go.html

Demo and classwork

Create a new WebAPI template and show that it works.

In the YAML, anything NOT in a list (denoted by '-') runs in parallel.

https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?view=azure-devops&tabs=schema%2Cparameter-schema#triggersazure pipelines

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

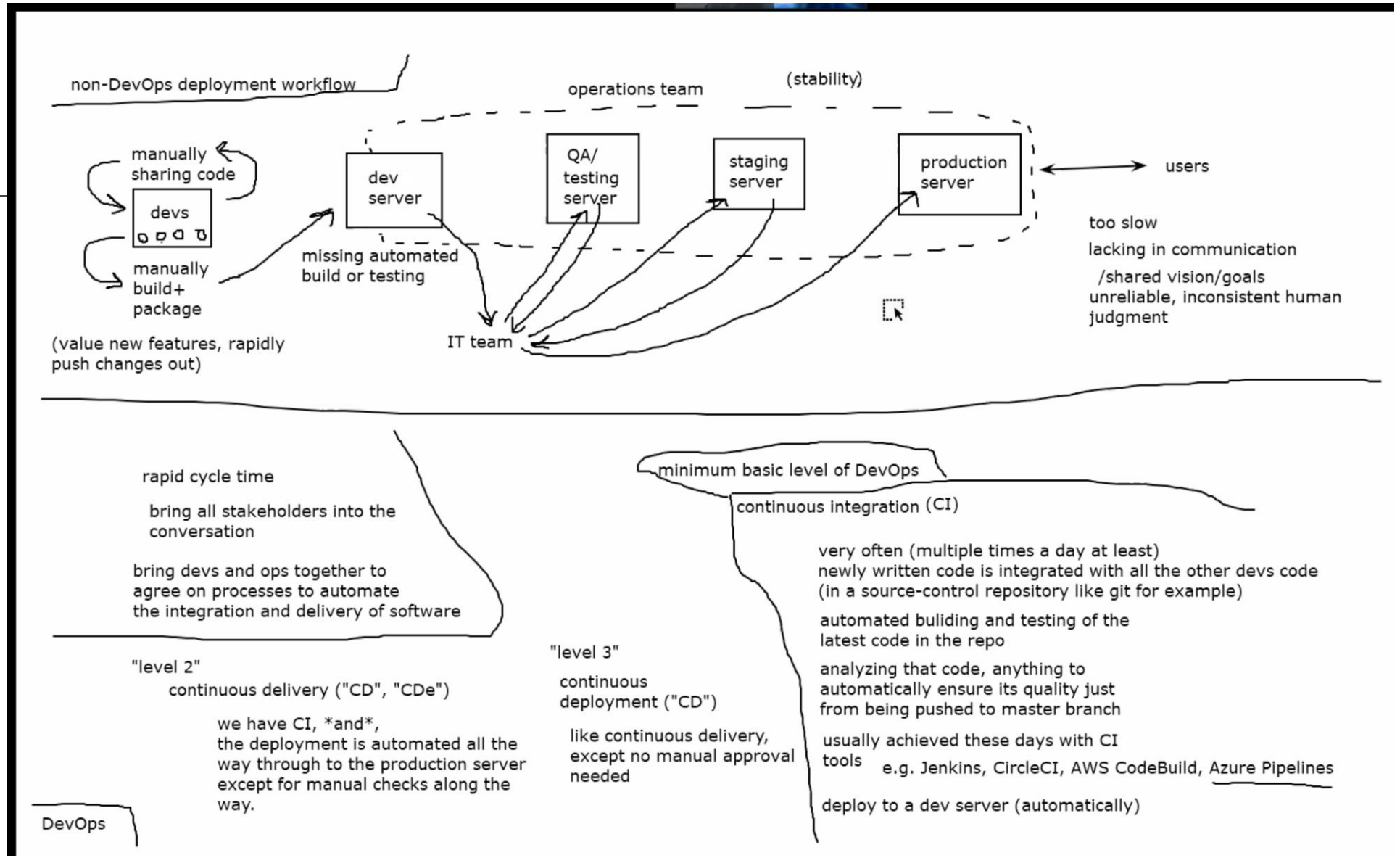
Azure Storage

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
operations team
(stability)
manually
sharing code
devs
manually
build+
package
(value new features, rapidly
push changes out)
dev
server
missing automated
build or testing
QA/
testing
server
staging
server
production
server
users
too slow
lacking in communication
/shared vision/goals
unreliable, inconsistent human
judgment
IT team
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
"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.
DevOps
"level 3"
continuous
deployment ("CD")
like continuous delivery,
except no manual approval
needed
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)

Showing dev.azure.com

Go to dev.azure.com

- Log in
- Go to “get started” and make a new organization
- Create a git repo special for this project. It MUST have a MVC project I just did the default sample hello world project
- On Azure go to Pipelines and choose your recently created repo.
- Nick had to invite everyone to the batch organization so that they could see their repo.
- Complete the → Connect, select, configure, review tabs
- Explains the YAML and what YAML is…
  - A YAML defines a set of jobs that will run when triggered on a cloned version of your repo.
  - A job is a set of steps
  - By default , ALL branches get built.
  - Adding a trigger configuration like what’s in a YAML can tell it “only build main branch” (continue on next page.)

- A pool is a set of machines for running this job.
  - This configuration says “use Microsoft’s hosted Ubuntu Linux virtual machines”
  - We could (if was wanted) have the YAML connect to a machine WE control to run the job on.
- Steps
  - This is the steps for the job
  - “script” is one kind of step (it’s a subsection of steps.) – on Linux run in bash
  - On windows it’ll run in command prompt
  - Make sure to give all steps a good display name.
  - Hint--- after steps:’ place a ‘|’ (this is in the sample YAML)
  - Sometimes you need quotes in the YAML

Nick makes a sample YAML (try to get an image of it!)

```yaml
# Azure Pipelines YAML documentation
# https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema

# this file defines a set of jobs that will run when triggered on a cloned version of your repo.
# a job is a set of steps
# by default, ther'es just one job, and you give it its list of steps

# by default, all branches get built
# but adding a trigger configuration like this says "only build master branch"
trigger:
- master

# a pool is a set of machines for running this job.
# this configuration says: "use Microsoft's hosted Ubuntu Linux virtual machines"
# (we could, if we wanted, have it connect to a machine WE control to run the job on)
pool:
  vmImage: 'ubuntu-latest'

# the steps in the job
# "script" is one kind of step - on linux it'll run in bash
# on windows it'll run in command prompt
# give all steps a good display name
steps:
- script: echo Hello, world!
  displayName: Print hello world

- script: |
    echo "Showing current directory contents"
    ls
  displayName: Show directory contents
```

Adding code coverage to the pipeline

Practice creating and making the YAML work with Github!

Use coverlet.. Sonarcloud only accepts opencover.

Look at his azure-pipelines.yaml in restaurant reviews.

```
    --configuration $(buildConfiguration)
  displayName: dotnet build

- script: dotnet test $(solutionPath)
    --configuration $(buildConfiguration)
    --collect "XPlat Code Coverage"
    --logger trx
    --no-build
    --results-directory $(Common.TestResultsDirectory)
    --settings coverlet.runsettings
  continueOnError: true
  displayName: dotnet test
  escalonn, 12 days ago • Update azure-pipelines.yml for Azure Pipelines
- task: SonarCloudAnalyze@1
  displayName: sonarcloud analysis run

- task: PublishTestResults@2
  condition: succeededOrFailed()
```