DevOps

Week 01

Murtaza Munawar Fazal

Class Rules

1

Attendance will be taken at the start of the class. Anyone coming after the attendance would be marked as Late. No leniency in attendance.

2

Quizzes will be unannounced with no retakes.

3

After every quiz/assignment, marks will be uploaded on google classroom. Make sure you verify all your marks.

4

Keep your cell phones on silent.

5

Copied Assignments will be graded 0.

Consultation Hours

- Available only on class day(s)
  - As per timetable version August 22, 2023:
    - Tuesday : 12:00 pm – 03:55 pm
    - Wednesday : 09:00 am – 01:00 pm

- For any queries after the consultation hours, please don't hesitate to send me an email : murtaza.fazal.v@nu.edu.pk

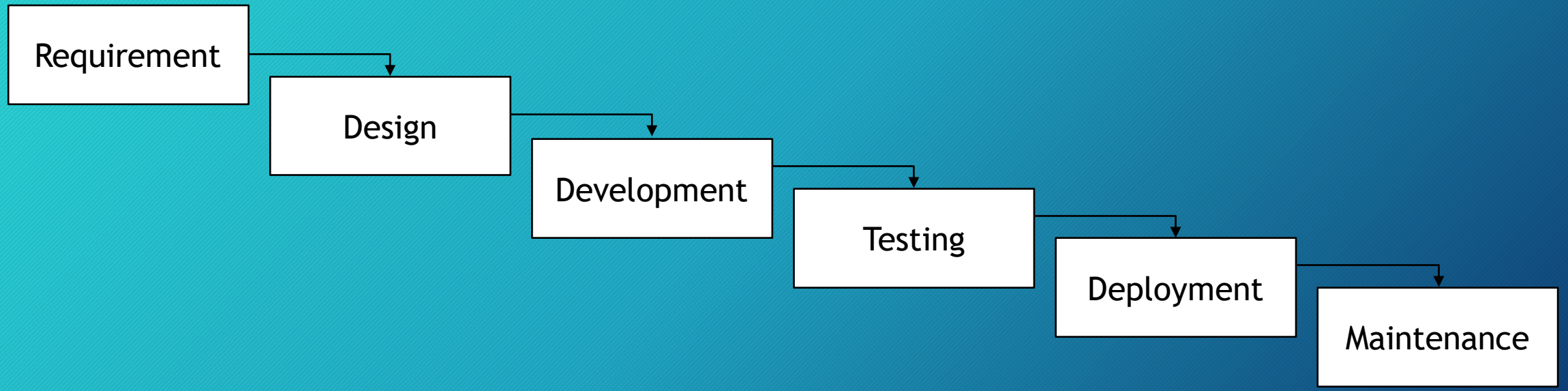
Tools Required

- Visual Studio (Code or 2019 Enterprise)
- Azure Account
- Github Account

Differentiate

- Software Developer v/s Software Tester
- Software Developer vs Software Engineer

Software Development Life Cycle (SDLC)



Waterfall

Software Development Life Cycle (SDLC)

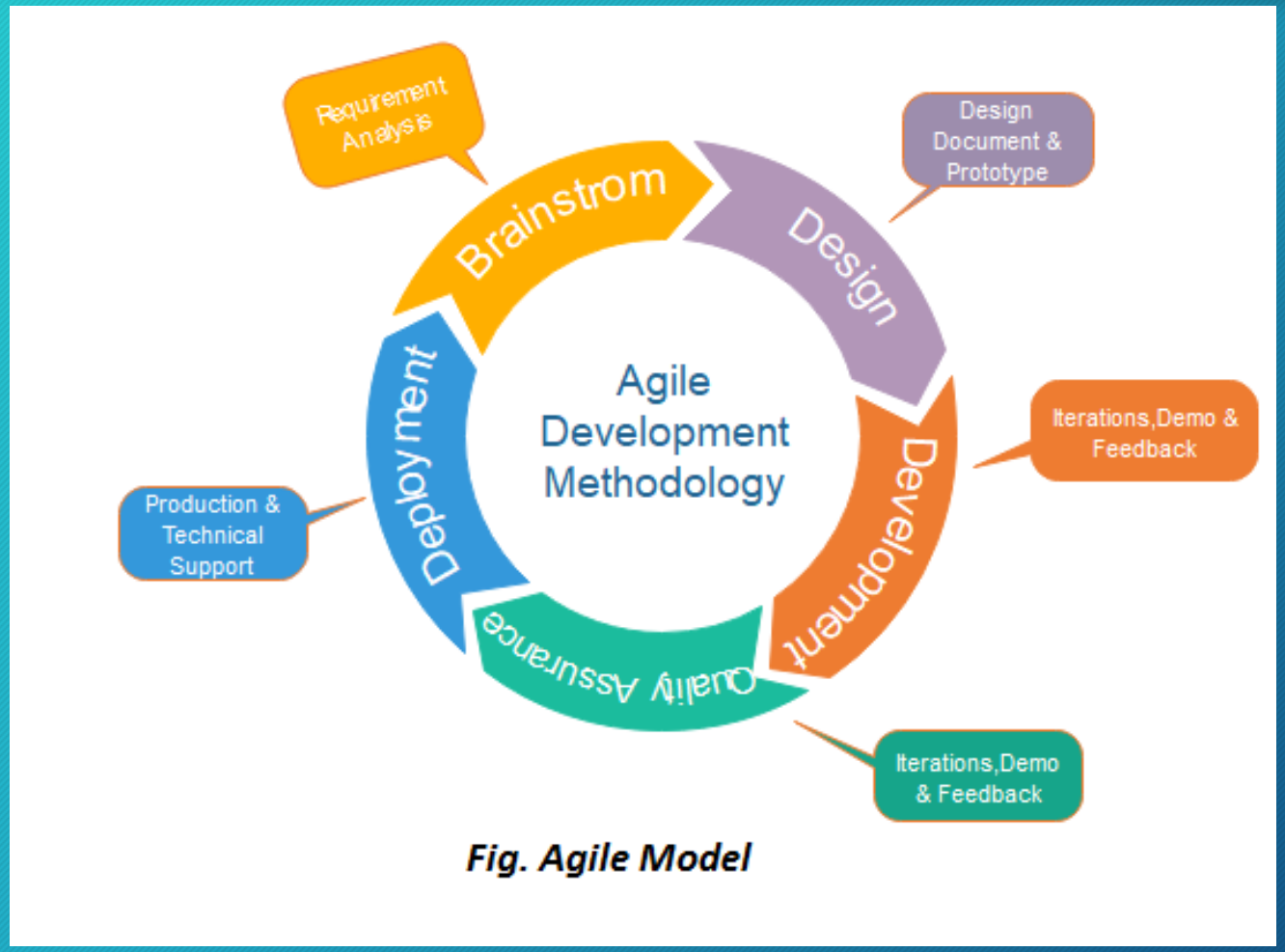


Fig. Agile Model

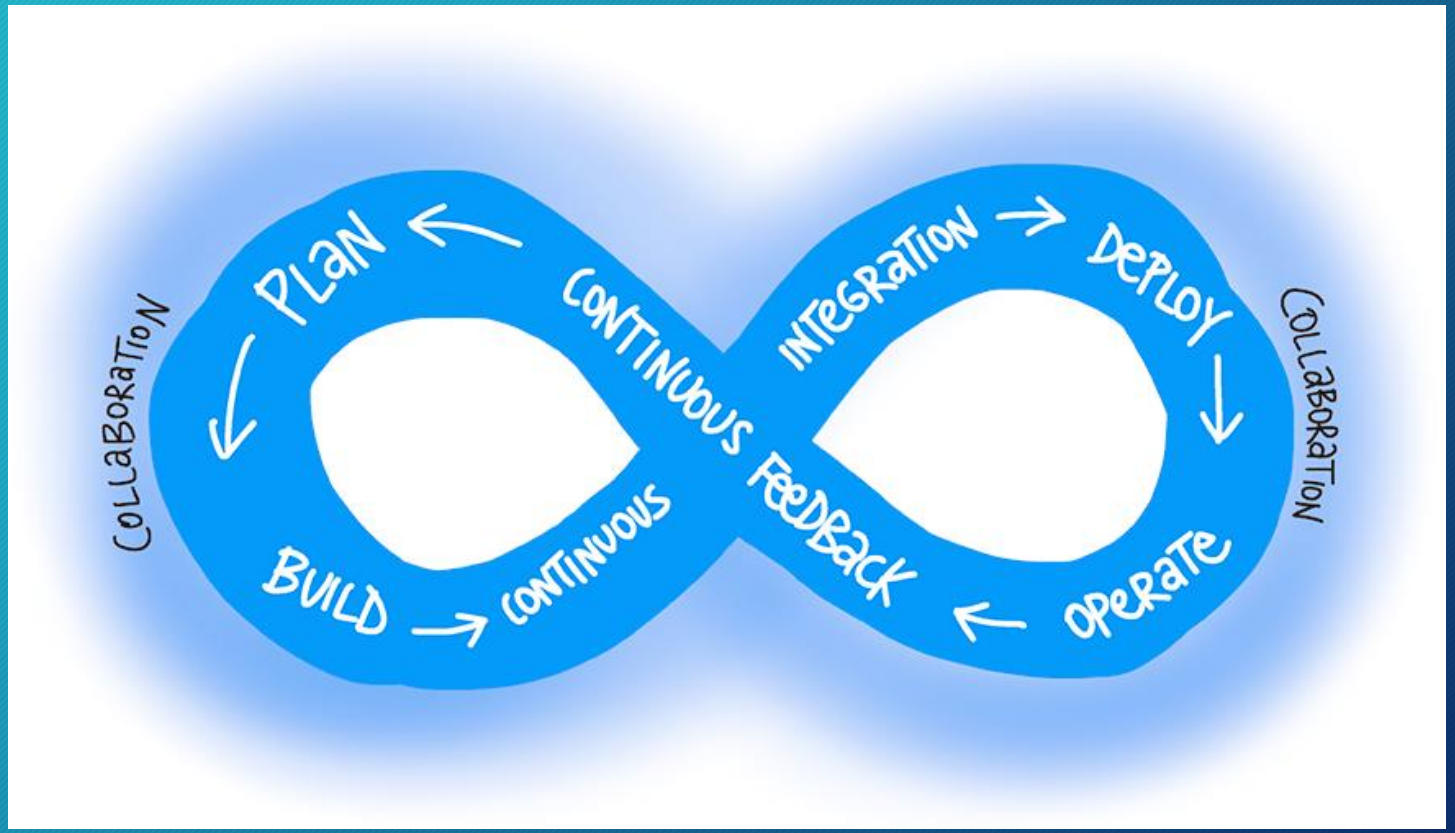
Agile

- Agile methodology constantly emphasizes adaptive planning and early delivery with continual improvement.
- Agile software development methods are based on releases and iterations:
  - One release might consist of several iterations.
  - Each iteration is like a small independent project.
  - After being estimated and prioritization:
    - Features, bug fixes, enhancements, and refactoring width are assigned to a release.
    - And then assigned again to a specific iteration within the release, generally on a priority basis.
  - At the end of each iteration, there should be tested working code.
  - In each iteration, the team must focus on the outcomes of the previous iteration and learn from them.
- Having teams focused on shorter-term outcomes is that teams are also less likely to waste time over-engineering features. Or allowing unnecessary scope creep to occur.

What is DevOps

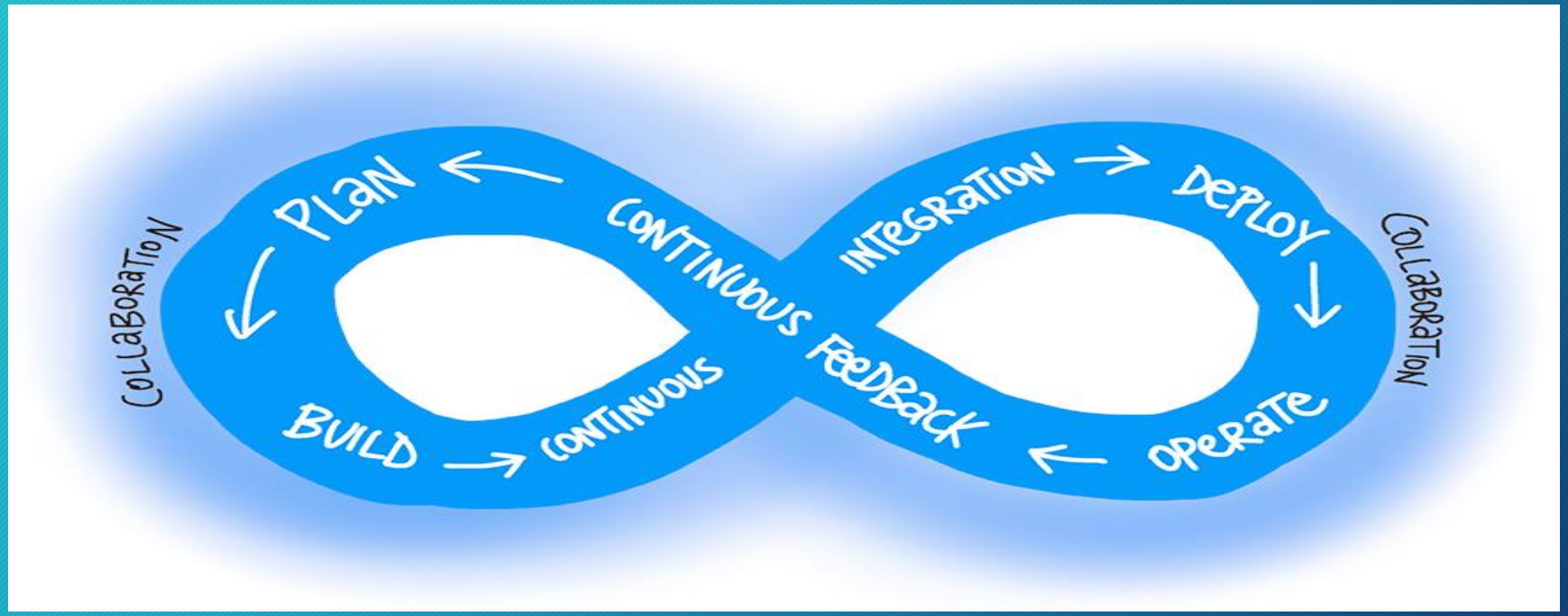
- *"DevOps is the union of people, process, and products to enable continuous delivery of value to end users."*

– Donovan Brown



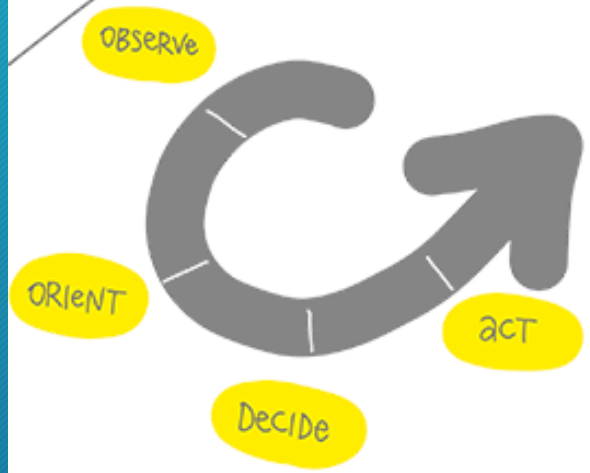
What is DevOps

- The contraction of "Dev" and "Ops" refers to replacing siloed Development and Operations. The idea is to create multidisciplinary teams that now work together with shared and efficient practices and tools. Essential DevOps practices include agile planning, continuous integration, continuous delivery, and monitoring of applications. DevOps is a constant journey.



Understand the Cycle

- You start with observing business, market, needs, current user behavior, and available telemetry data. Then you orient with the enumeration of options for what you can deliver, perhaps with experiments. Next, you decide what to pursue, and you act by delivering working software to real users.

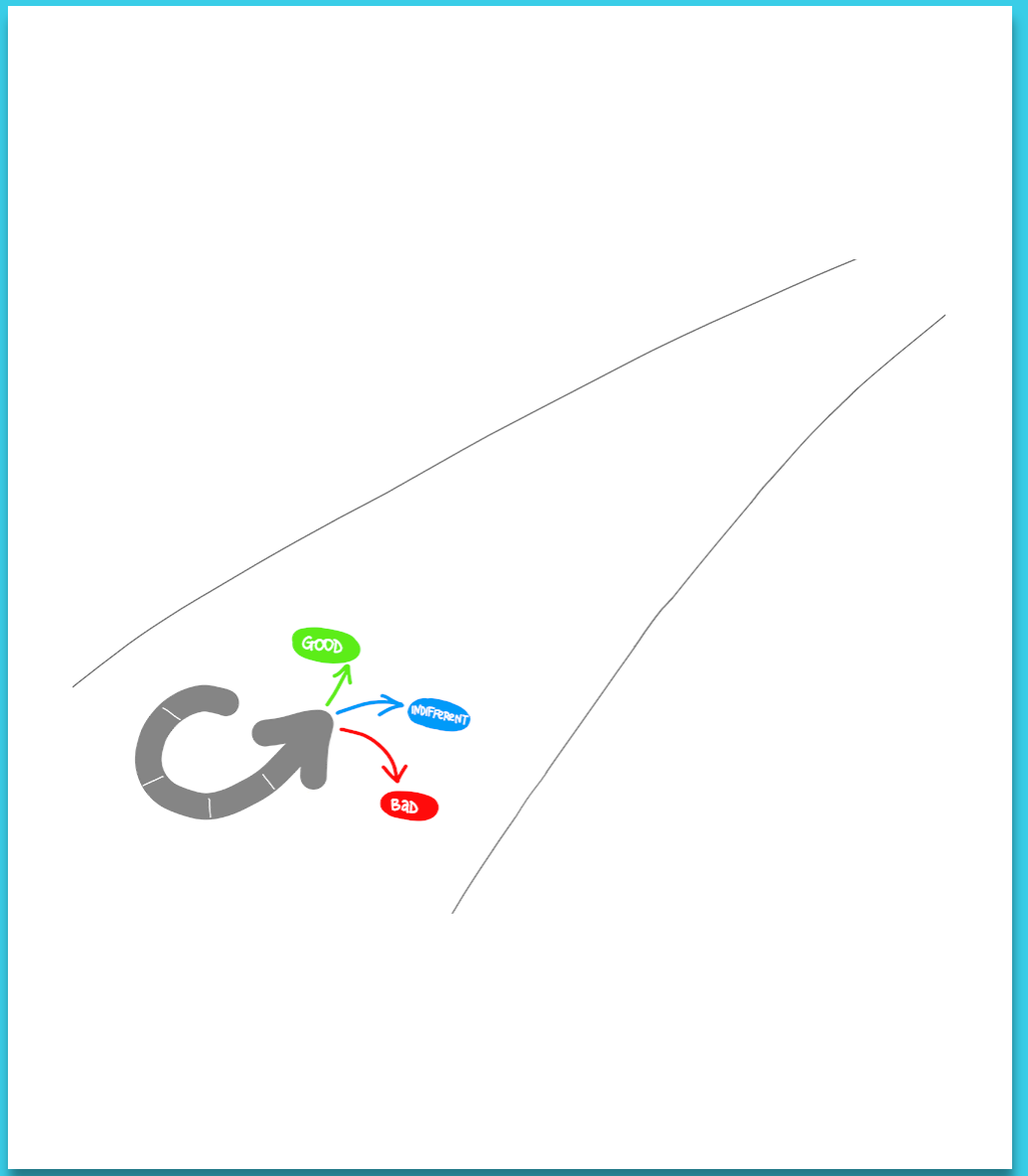


Understand the Cycle

- Many experience reports tell us that roughly one-third of the deployments will have negative business results. Approximately one-third will have positive results, and one-third will make no difference. Fail fast on effects that do not advance the business and double down on outcomes that support the business.

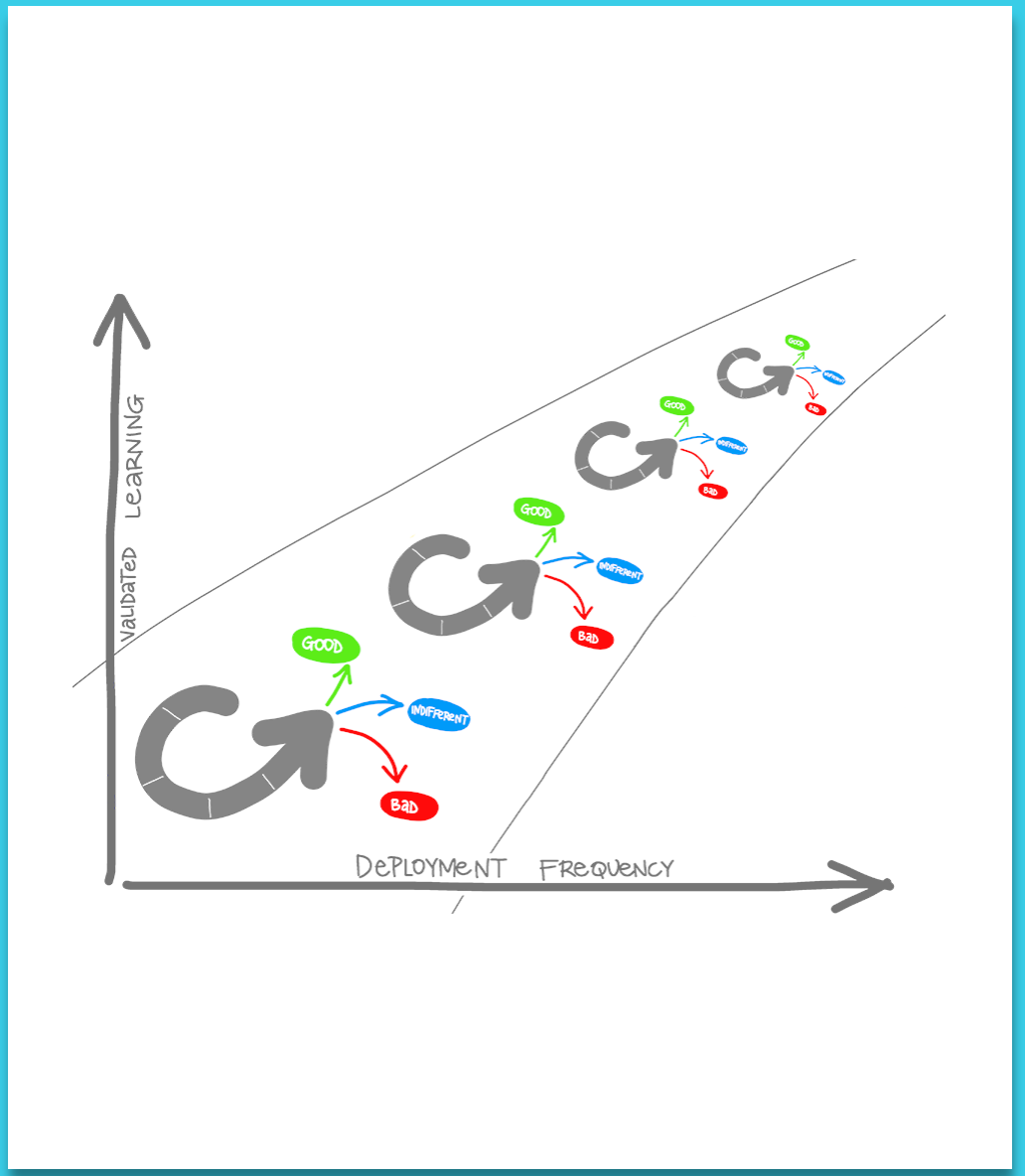
Validate Learning

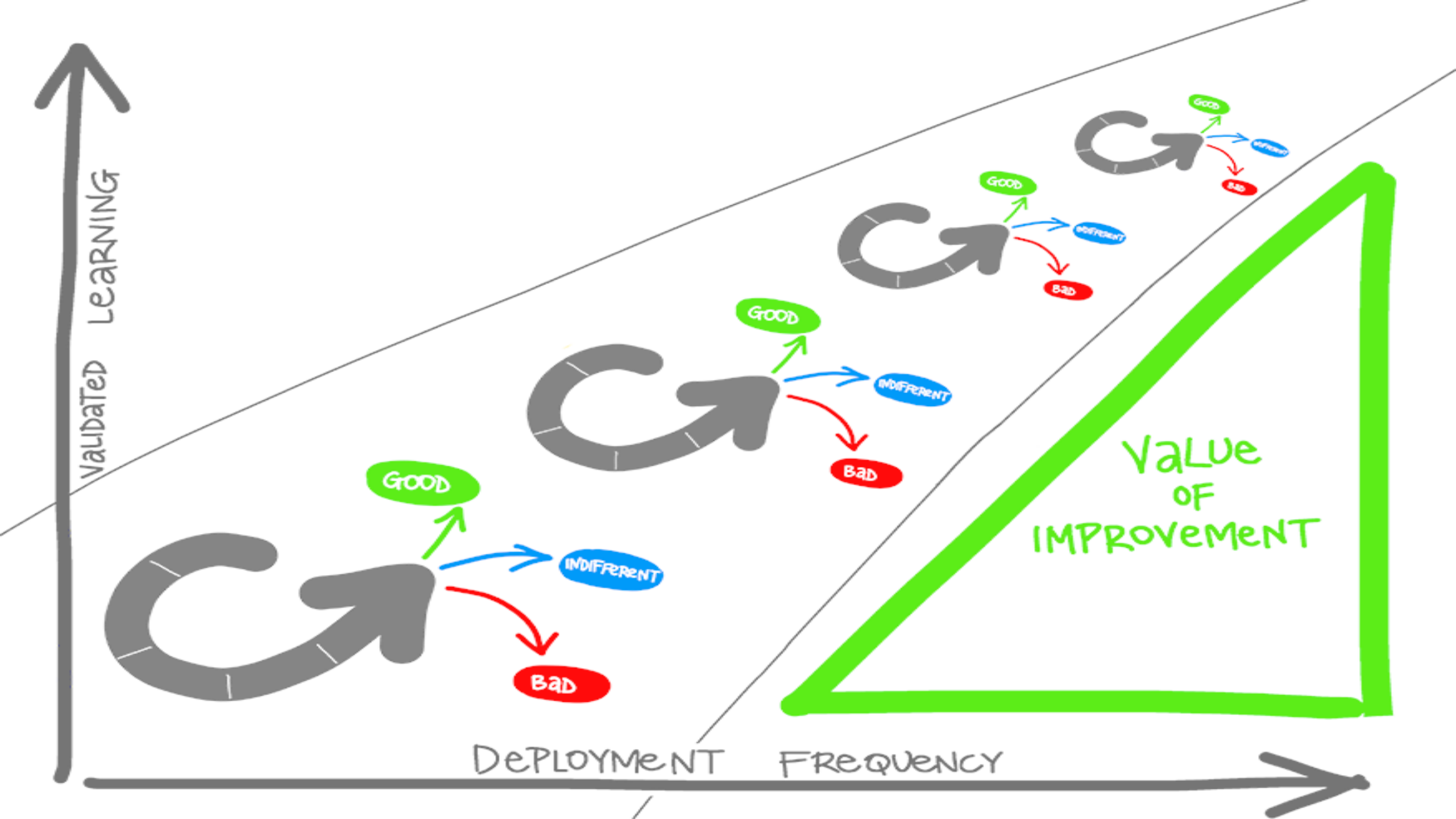
- How quickly you can fail fast or double down is determined by your cycle time. Also, in how long that loop takes, or in lean terms. Your cycle time determines how quickly you can gather feedback to determine what happens in the next loop. The feedback that you collect with each cycle should be factual, actionable data.



DevOps Practices

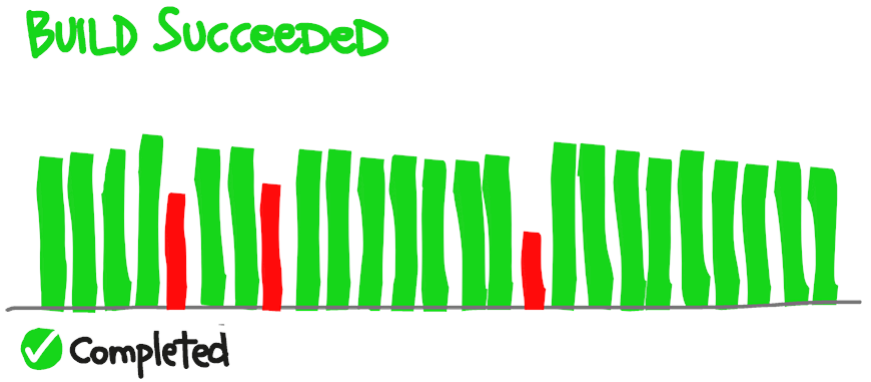
- You shorten your cycle time by working in smaller batches.
- Using more automation.
- Hardening your release pipeline.
- Improving your telemetry.
- Deploying more frequently.




VALIDATED LEARNING
GOOD
INDIFFERENT
BAD
GOOD
INDIFFERENT
BAD
GOOD
INDIFFERENT
BAD
GOOD
INDIFFERENT
BAD
VALUE OF IMPROVEMENT
DEPLOYMENT FREQUENCY

Continuous Integration

- How long does it take to deploy a change of one line of code or configuration?
  - Continuous Integration drives the ongoing merging and testing of code, leading to an early finding of defects. Other benefits include less time wasted fighting merge issues and rapid feedback for development teams.

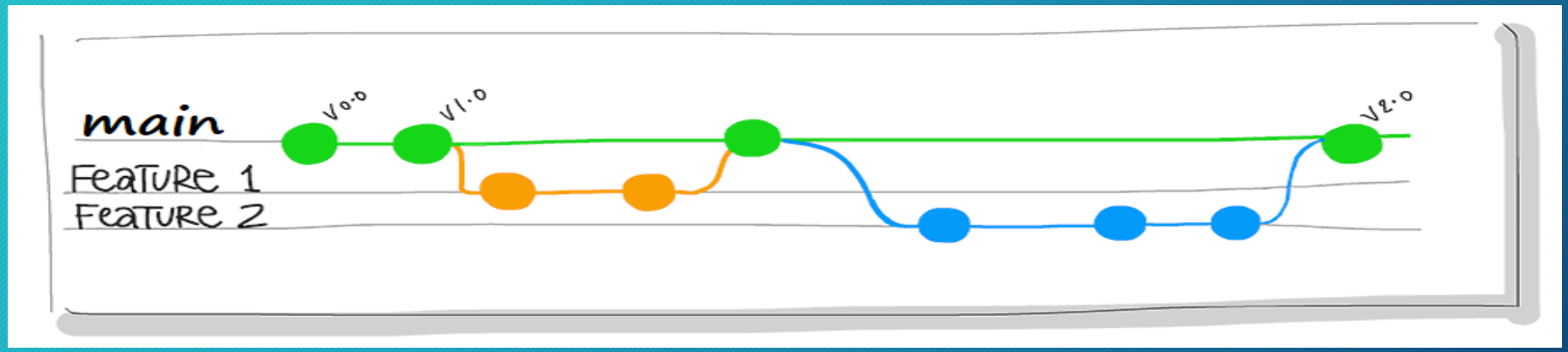


Continuous Delivery

- Continuous Delivery of software solutions to production and testing environments helps organizations quickly fix bugs and respond to ever-changing business requirements.

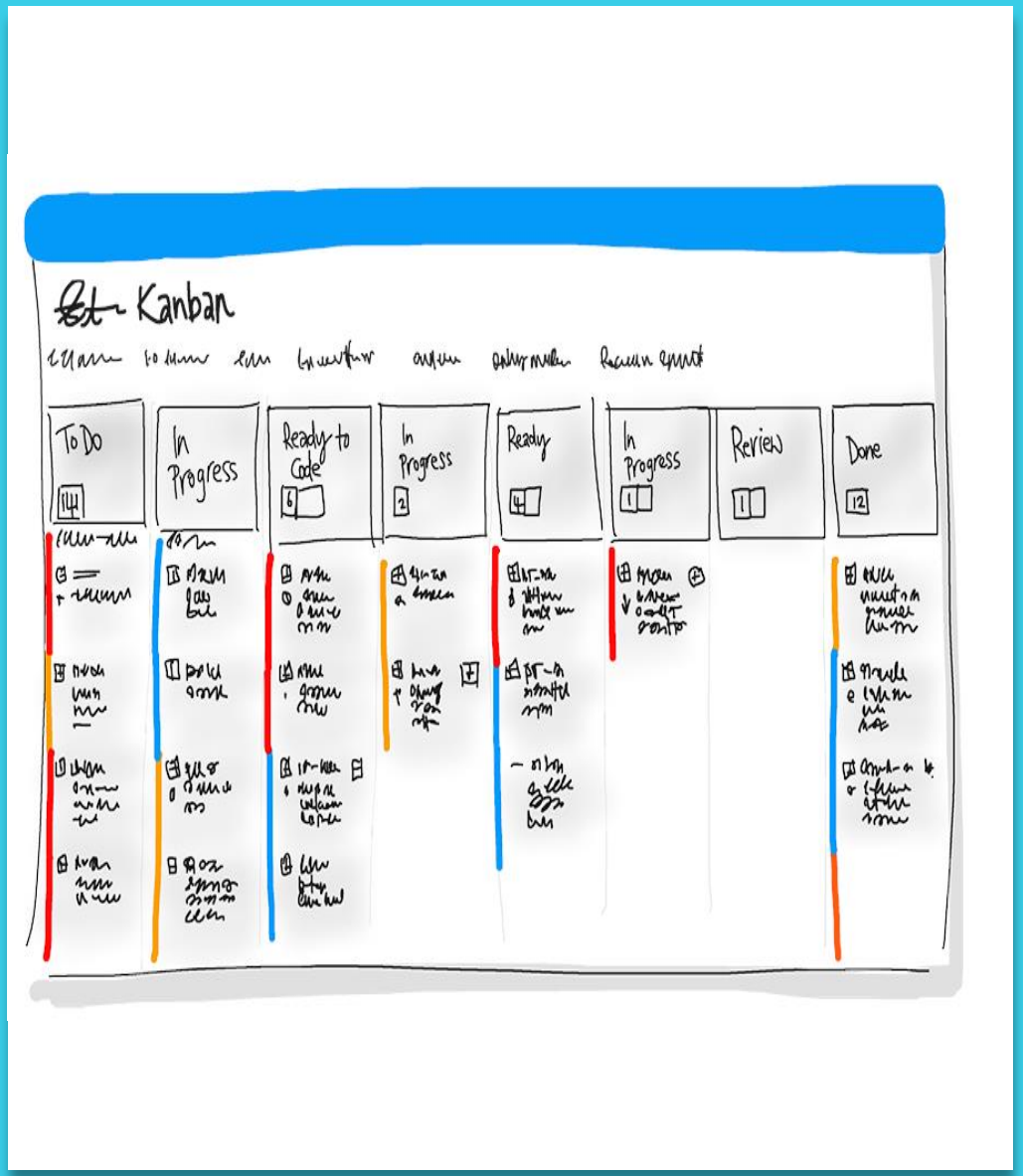
Version Control

- Version Control, usually with a Git-based Repository, enables teams worldwide to communicate effectively during daily development activities. Also, integrate with software development tools for monitoring activities such as deployments.



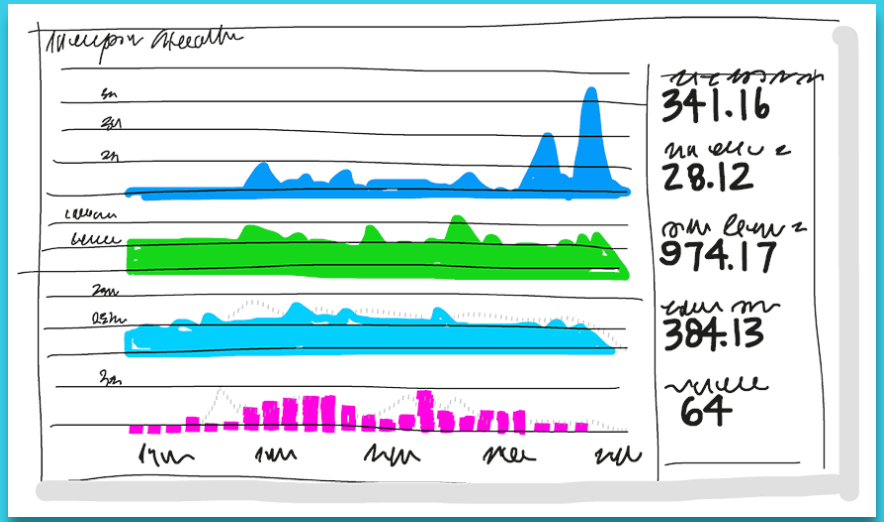
Agile Planning

- Use Agile planning and lean project management techniques to:
  - Plan and isolate work into sprints.
  - Manage team capacity and help teams quickly adapt to changing business needs.
  - A DevOps Definition of Done is working software collecting telemetry against the intended business goals.



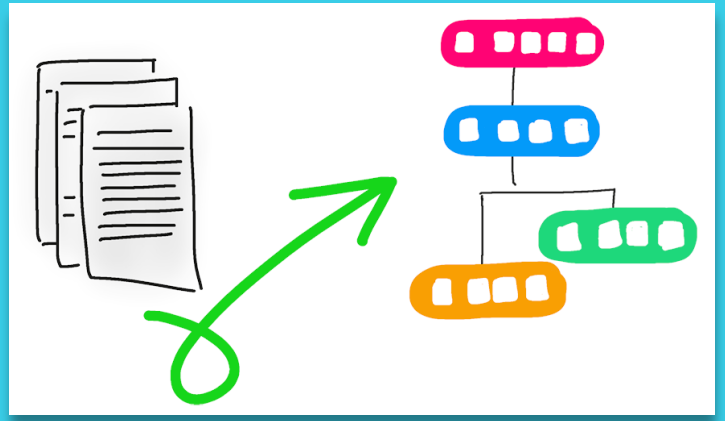
Monitoring and Logging

- Monitoring and Logging of running applications. Including production environments for application health and customer usage. It helps organizations create a hypothesis and quickly validate or disprove strategies. Rich data is captured and stored in various logging formats.



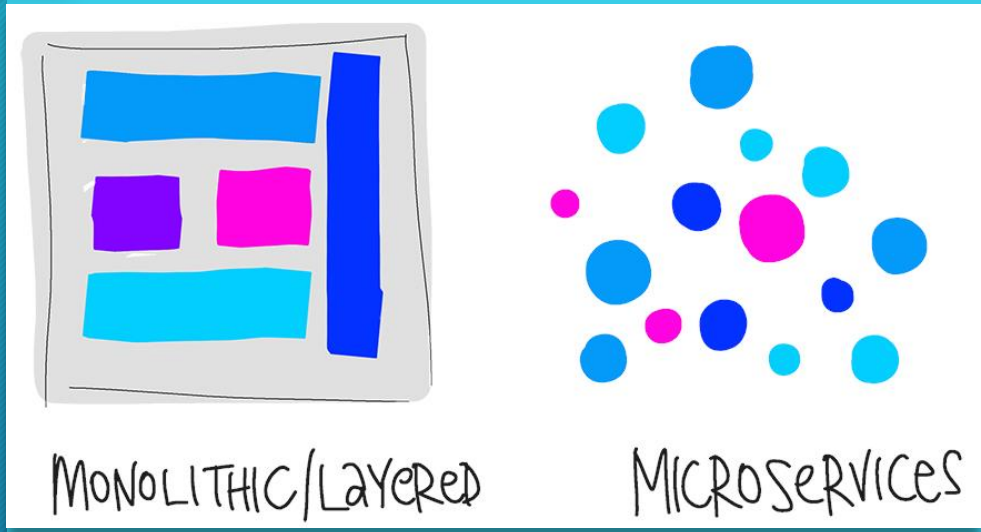
Infrastructure as Code (IaC)

- Enables the automation and validation of the creation and teardown of environments to help deliver secure and stable application hosting platforms.



Microservices architecture / Containers

- Use Microservices architecture to isolate business use cases into small reusable services that communicate via interface contracts. This architecture enables scalability and efficiency.
- Containers are the next evolution in virtualization. They're much more lightweight than virtual machines, allow much faster hydration, and easily configure files.



Apply DevOps to which Project ?

- The terms greenfield and brownfield have their origins in residential and industrial building projects.
- A greenfield project is one done on a green field, undeveloped land.
- A brownfield project is done on the used ground for other purposes. Because of the land use that has once occurred, there could be challenges reusing the land. Like existing buildings, some would be obvious but less obvious, like polluted soil.

Apply DevOps to which Project ?

- Greenfield Project:
  - A greenfield project will always appear to be a more accessible starting point. A blank slate offers the chance to implement everything the way that you want.
  - You might also have a better chance of avoiding existing business processes that do not align with your project plans.
  - Suppose current IT policies do not allow the use of cloud-based infrastructure. In that case, the project might be qualified for entirely new applications designed for that environment from scratch.
  - For example, you can sidestep internal political issues that are well entrenched.

Apply DevOps to which Project ?

- Brownfield Project:
  - Usually, brownfield projects come with:
    - The baggage of existing codebases.
    - Existing teams.
    - A significant amount of technical debt.

Apply DevOps to which Project ?

- Greenfield DevOps project would be easier to manage and to achieve success?
- A common misconception is that DevOps is only for greenfield projects and suits startups best. However, DevOps can also succeed with brownfield projects.
- The beauty of these projects is that there's often a large gap between customer expectations and delivery.
- The teams involved may well realize that the status quo needs to change. They've lived the challenges and the limitations associated with what they're currently doing.
- The system is often crucial for organizations. It might also be easier to gain more robust management buy-in for these projects because of the potential benefits delivered.
- Management might also have a stronger sense of urgency to point brownfield projects in an appropriate direction when compared to greenfield projects that do not currently exist.

Types of System

- Systems of record
  - Systems that provide the truth about data elements are often-called systems of record. These systems have historically evolved slowly and carefully. For example, it is crucial that a banking system accurately reflects your bank balance. Systems of record emphasize accuracy and security.
- Systems of engagement
  - Many organizations have other systems that are more exploratory. These often use experimentation to solve new problems. Systems of engagement are modified regularly. Usually, it is a priority to make quick changes over ensuring that the changes are correct.

Types of Users

- Canary
  - **Canary** users voluntarily test bleeding edge features as soon as they're available.
- Early Adopters
  - **Early adopters** who voluntarily preview releases, considered more refined than the code that exposes canary users.
- Users
  - **Users** who consume the products after passing through canary and early adopters.

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

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	

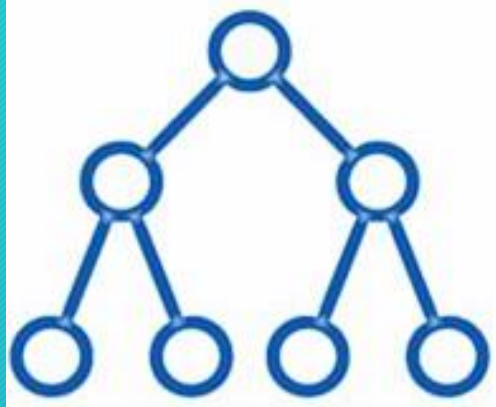
Centralized Source Control

- 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.

Distributed Source Control



Distributed

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.

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.

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.