DevOps

Modern Application Development and Deployment Process

Agenda

- DevOps Concepts and Tools
- DevOps stages and tools
- Version Control Systems
- Software Build Tools / Maven
- Testing Software
- Containers / Docker
- Continuous Integration using Jenkins
- Continuous Testing using Selenium
- Configuration Management using Ansible
- Continuous Monitoring
- Configuration Management with Puppet
- Containerization using Kubernetes
- IaC with Terraform

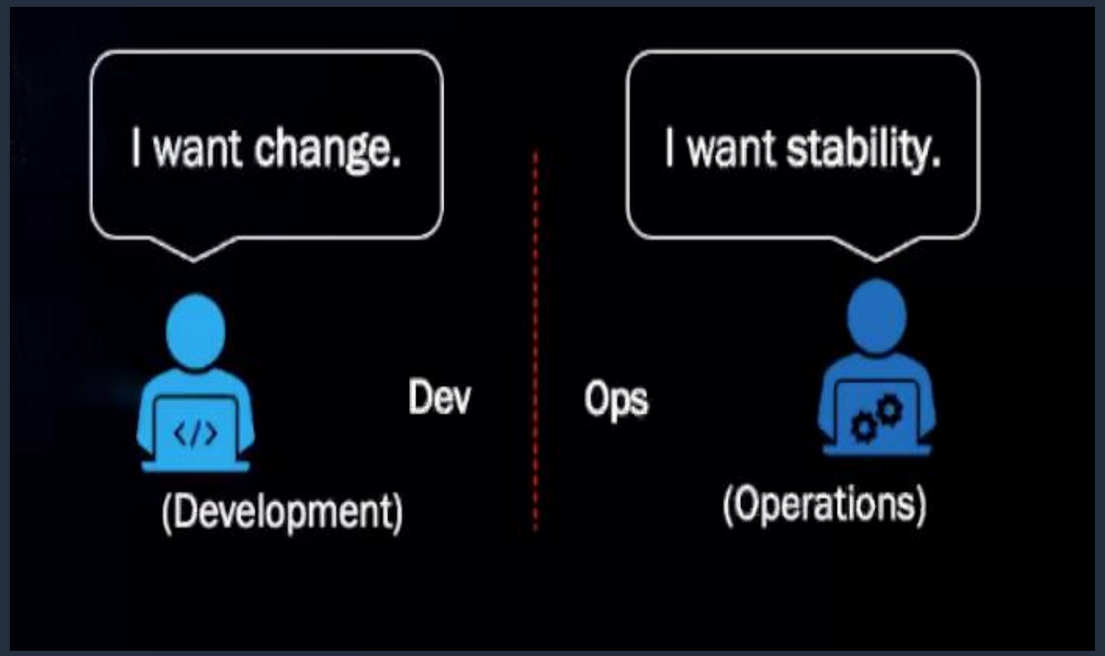
ToolSets

Tools	Learn	Tools	Learn
Github, BitBucket		Microservices, SpringBoot Java	
Docker containter		Windows, Webshpere	
		BlackDuck, Fortify, SonarQube	
Technlogy(AWS/Azure/GCP)		Cloud Native	
Jenkins		IaC Terraform	
Jira, Confluence		SPlunk, Cassandra	
Artifactory		kubernetes	
Azure DevOps			
Powershell, Bash scripting			
AIX/Linux			

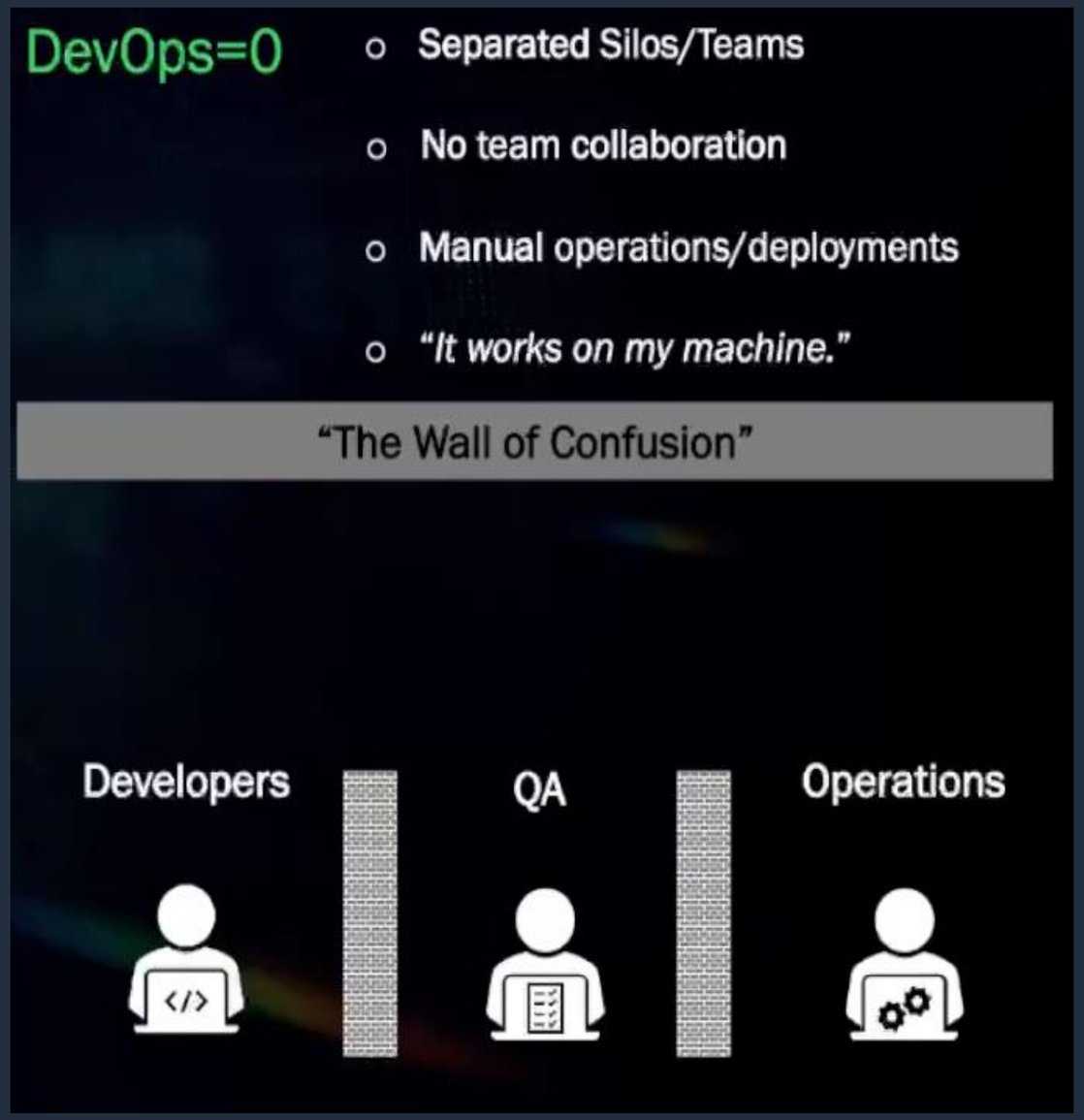
DevOps Concepts and Tools

What is DevOps

DevOps is a set of practices that combines software development (Dev) and IT operations (ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality. “ Wikipedia



With / Without DevOps

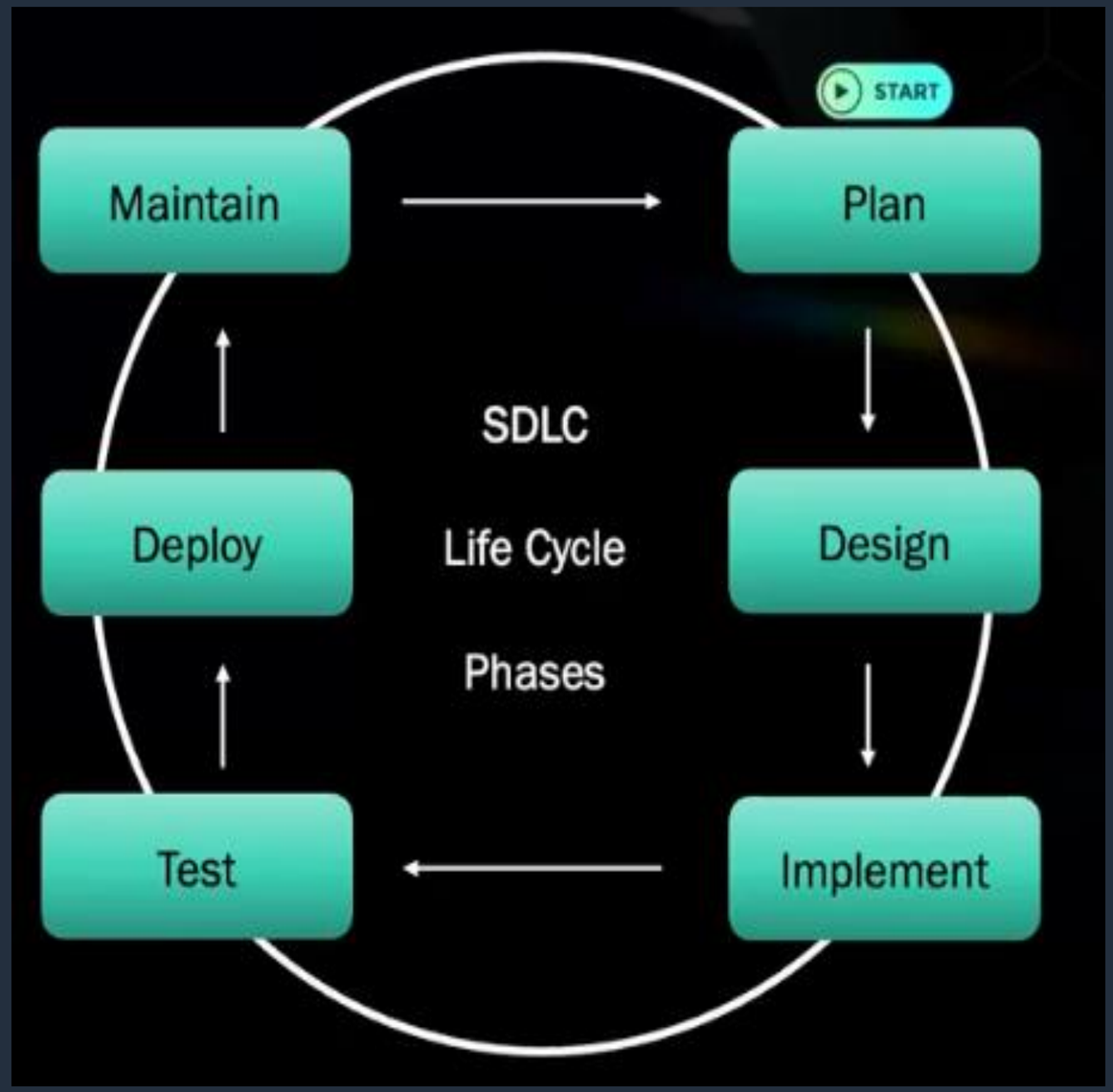


What is SDLC ?

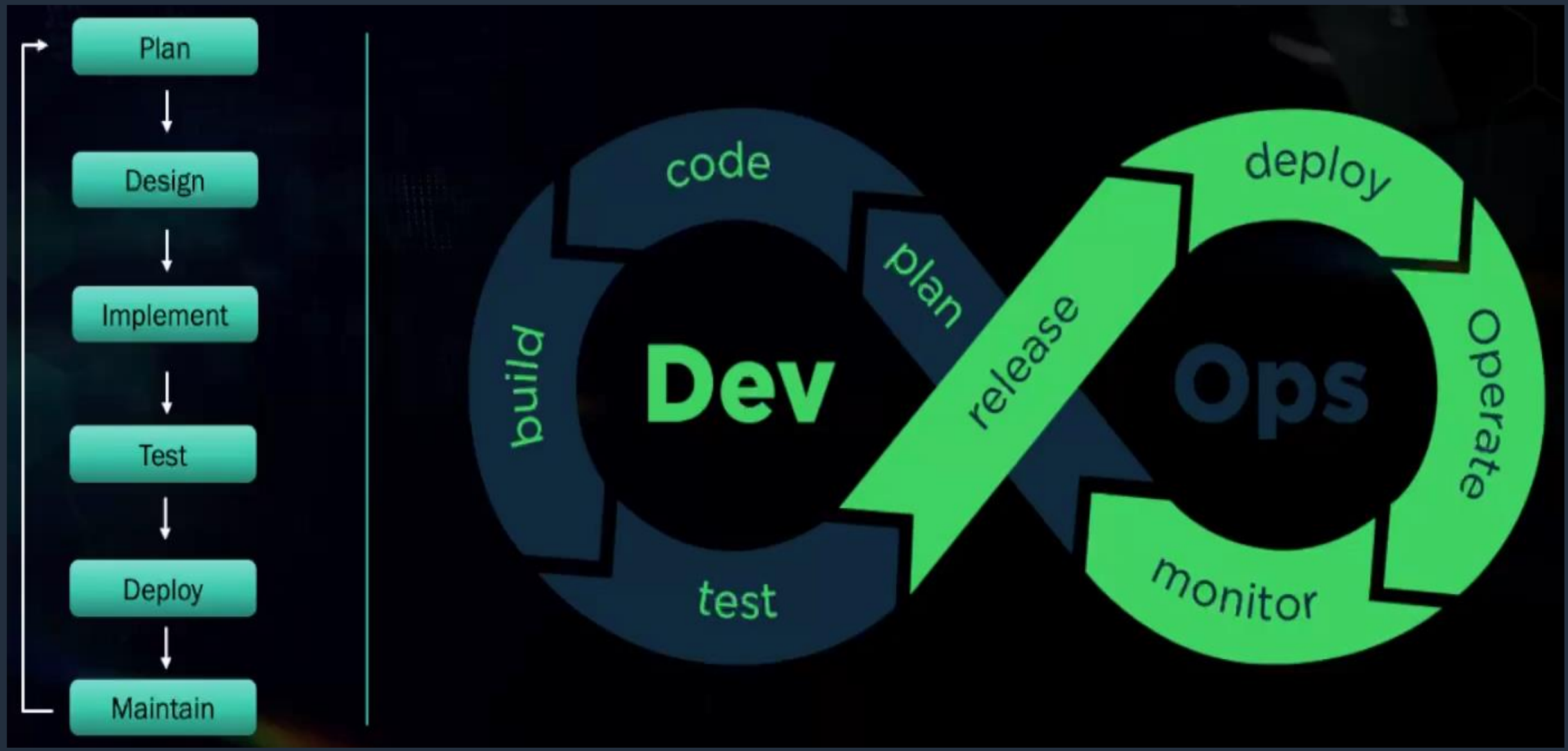
The Software development lifecycle (SDLC) is a framework to produce high-quality software in a systematic, time and cost-effective way.

It consists of a plan that describes how the software will be developed, maintained and improved.

Every phase of the SDLC life cycle has its own process and deliverables that feed into the next phase

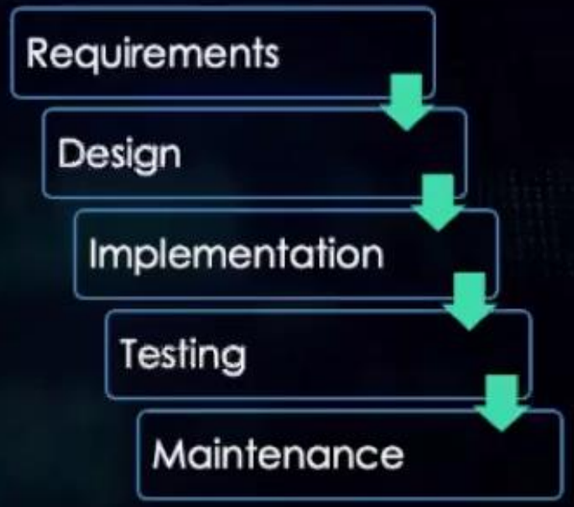


SDLC - DevOps


Plan
Design
Implement
Test
Deploy
Maintain
code
plan
build
test
Dev
release
deploy
operate
monitor
Ops

WATERFALL Vs. AGILE

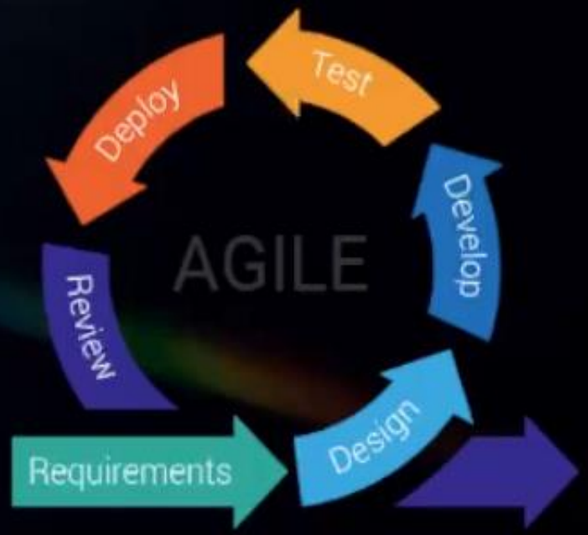
WATERFALL



Waterfall Model breaks down project activities into sequential order, and the project development team only moves to next phase if the previous step completed successfully.

"The subsequent phases rely on the deliverables of the previous one."

AGILE



Agile methodology divides the project into several stages, providing continuous iteration between development and testing in the software development process.

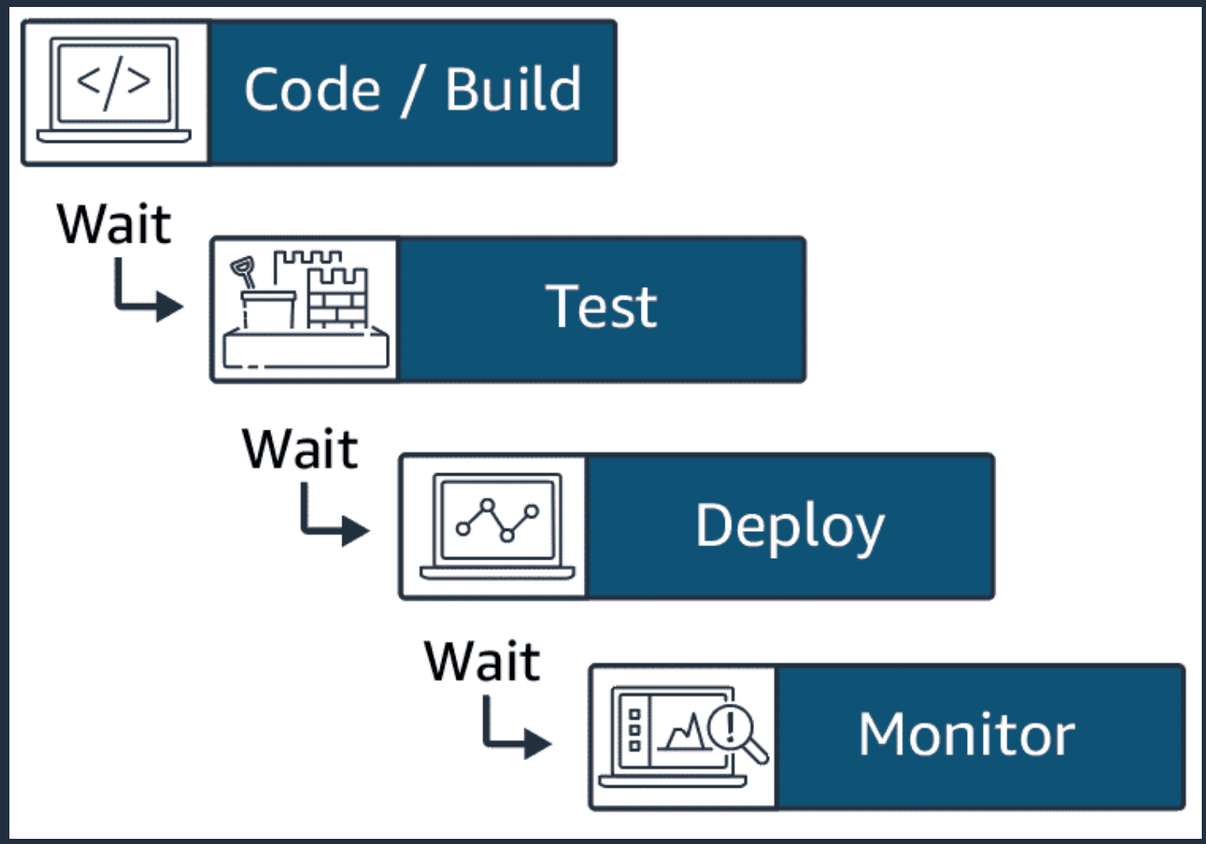
"This process allows more communication/collaboration between teams."

Problems with Traditional Development Practices

- Traditional ways of developing software have proven slow and inefficient, and fail to support teams' efforts to quickly deliver stable, high-quality applications.
- Waterfall development projects are slow, not iterative, resistant to change, and have long release cycles.

- Some reasons for this include:

- Requirements are rigid
- Development phases are siloed
- Hand offs from one phase to the other are long
- Testing and security come after implementation



Problems with Traditional Development Practices

- Traditional ways of developing software have proven slow and inefficient, and fail to support teams' efforts to quickly deliver stable, high-quality applications.
-
- Monolithic applications are hard to update and deploy because they:
  - Are developed and deployed as a unit
  - Have tightly coupled functionality,
  - Are implemented using a single development stack

Problems with Traditional Development Practices

- Teams supporting the software development lifecycle have traditionally been siloed.
- Specialized in their skill set, teams have been separated from each other
- Teams have common goals, still they also have their own priorities, tooling, and processes.
- It is difficult to achieve efficiencies when project members are reporting to different units and aimed for different targets.

Agile Software Development Process

- What is Agile
  - An iterative approach to project management and software development
  - helps teams deliver value to their customers faster and with fewer headaches.
  - Instead delivering Everything on a "big bang" launch vs delivers work in small, but consumable, increments.
  - Requirements, plans, and results are evaluated continuously for responding to change quickly.

Agile Software Development Process

- Manifesto for Agile Software Development

- We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

    - *Individuals and interactions* over *processes and tools*
    - *Working software* over *comprehensive documentation*
    - *Customer collaboration* over *contract negotiation*
    - *Responding to change* over *following a plan*

- That is, while there is value in the items on the right, we value the items on the left more.

Agile Software Development Process

- 12 Principals Behind Agile Manifesto

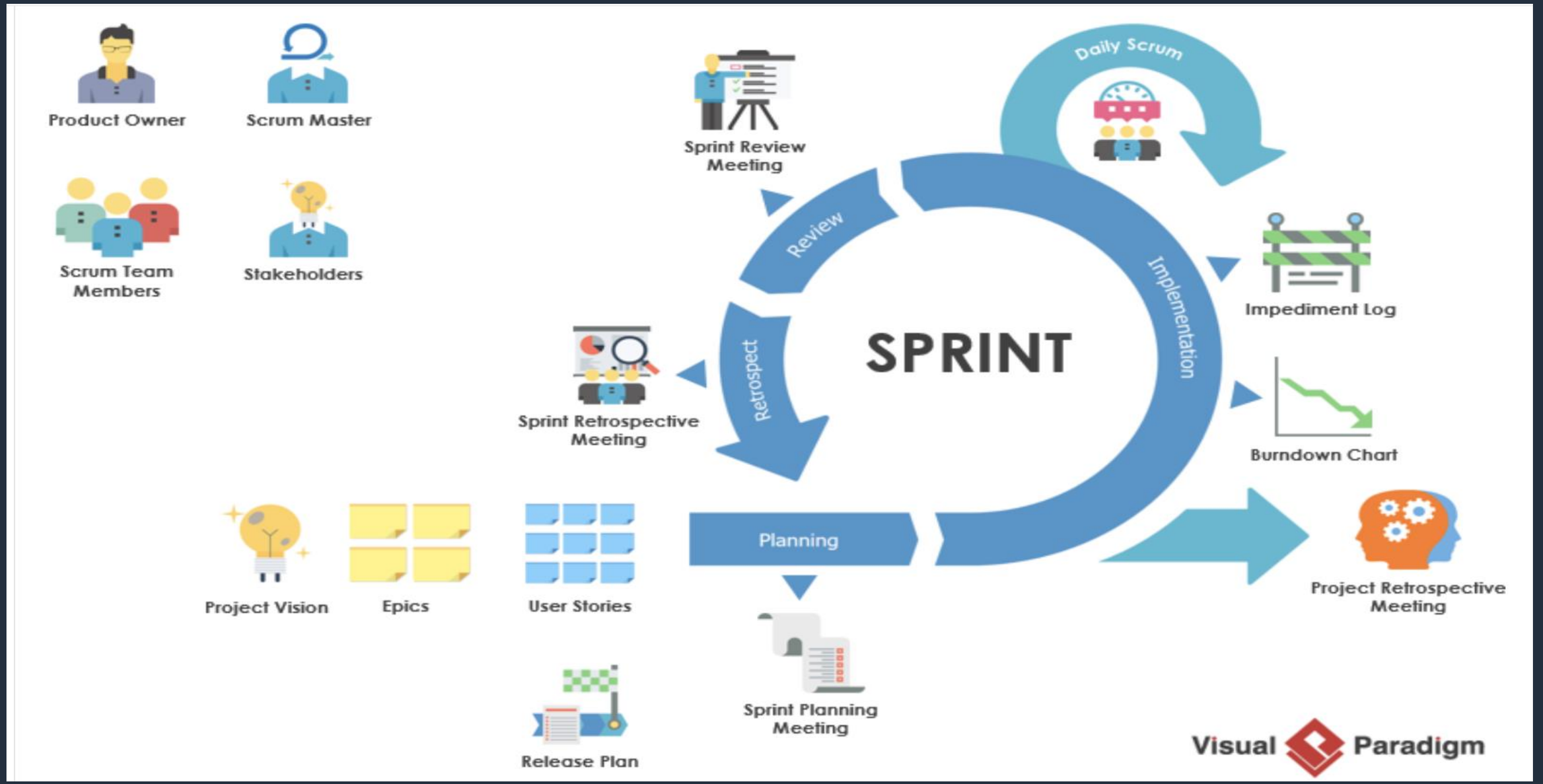
1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

Agile Software Development Process

- 12 Principals Behind Agile Manifesto

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The Agile Scrum Framework



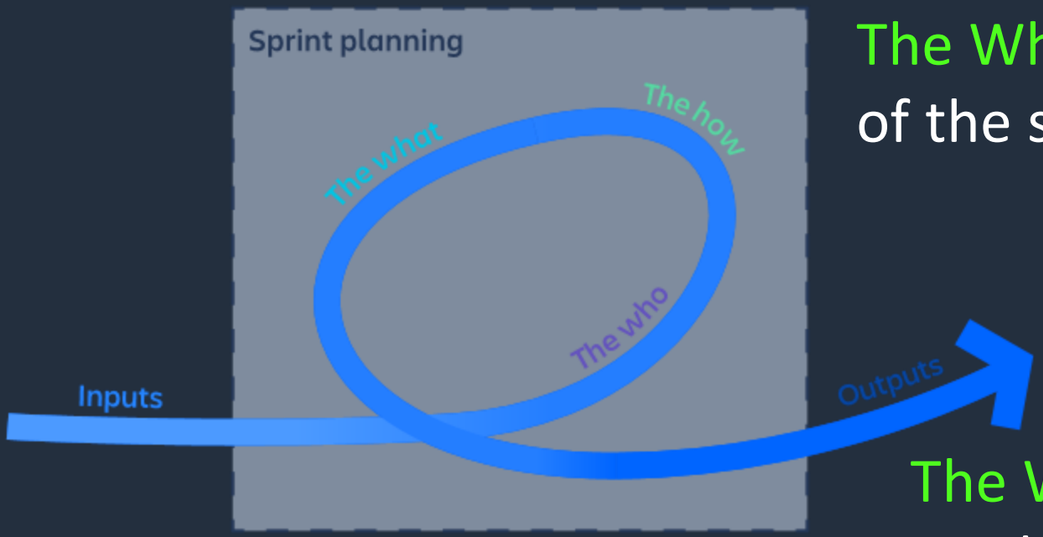
What and Why - SCRUM

- SCRUM is a framework that helps teams work together.
- Much like a rugby team (where it gets its name) training for the big game, scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.
- With scrum, a product is built in a series of iterations called SPRINTS that break down big, complex projects into bite-sized pieces
- A SPRINT is a short, time-boxed period when a scrum team works to complete a set amount of work.

"Sprints make projects more manageable, allow teams to ship high-quality work faster and more frequently, and gives them more flexibility to adapt to change."

What and Why - SCRUM

- SPRINT PLANNING is an event in SCRUM that kicks off the sprint.
- The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved.
- Sprint planning is done in collaboration with the whole scrum team.



The What – The product owner describes the objective(or goal) of the sprint and what backlog items contribute to that goal.

The How – The development team plans the work necessary to deliver the sprint goal.

The Who – You cannot do sprint planning without the product owner or the development team.

The Inputs – A great starting point for the sprint plan is the product backlog as it provides a list of ‘stuff’ that could potentially be part of the current sprint.

The Outputs – the team can describe the goal of the sprint and how it will start working toward that goal.

Sprint Ceremonies

DAILY SCRUM

Attendees: Development team, scrum master, product owner
When: Once per Daily (Morning)
Duration: No more than 15Mins
Purpose: quickly inform everyone of what's going on across the team.

SPRINT PLANNING

Attendees: Development team, scrum master, product owner
When: At the beginning of a sprint.
Duration: 1~2 Hours
Purpose: sets up the entire team for success throughout the sprint.

SPRINT REVIEW

Attendees: Development team, scrum master, product owner, Stakeholder
When: End of Sprint
Duration: Typically 60 Mins
Purpose: to showcase the work of the team.

Sprint Ceremonies

SPRINT PLANNING

Attendees: Development team, scrum master, product owner
When: At the beginning of a sprint.
Duration: 1~2 Hours
Purpose: sets up the entire team for success throughout the sprint.

SPRINT REVIEW

Attendees: Development team, scrum master, product owner, Stakeholder
When: End of Sprint
Duration: Typically 60 Mins
Purpose: to showcase the work of the team.

DAILY SCRUM

Attendees: Development team, scrum master, product owner
When: Once per Daily (Morning)
Duration: No more than 15Mins
Purpose: quickly inform everyone of what's going on across the team.

SPRINT RETROSPECTIVE

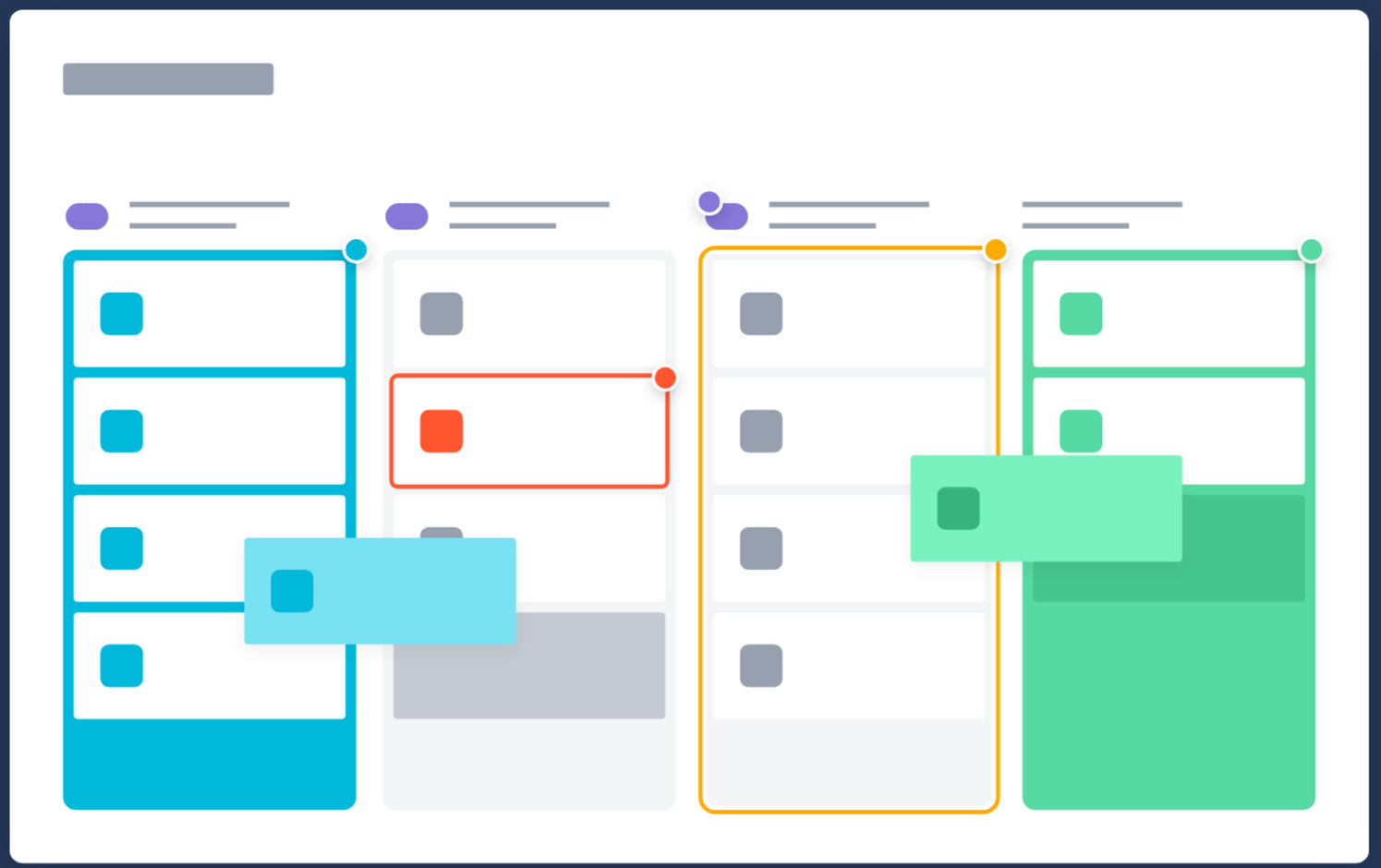
Attendees: Development team, scrum master, product owner
When: End of Sprint
Duration: Typically 45~60 Mins
Purpose: to get rapid feedback & understand what worked well–and what didn't.

What and Why – KANBAN?

- KANBAN is a popular framework used to implement Agile and DevOps Software Development.
- It requires real-time communication of capacity and full transparency of work.
- Work items are represented visually on a Kanban board, allowing team members to see the state of every piece of work at any time.

- A KANBAN BOARD is an agile project management tool designed to help
    - visualize work,
    - limit work-in-progress, and
    - maximize efficiency (or flow).

What and Why – KANBAN?

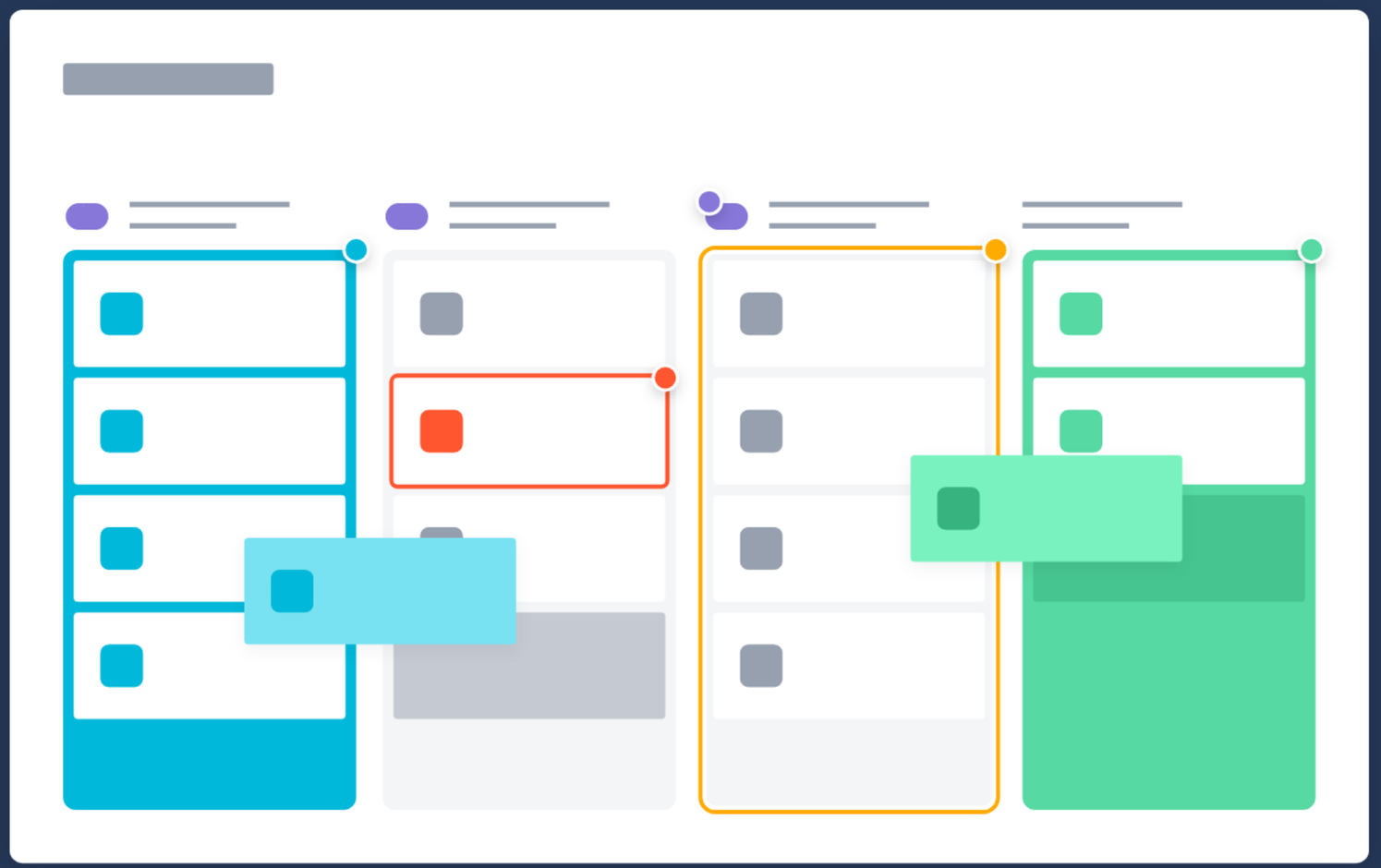


VISUAL SIGNALS help teammates and stakeholders quickly understand what the team is working on.

Each COLUMN represents a specific activity that together compose a “workflow”.

WIP LIMITS are the maximum number of cards that can be in one column at any given time.

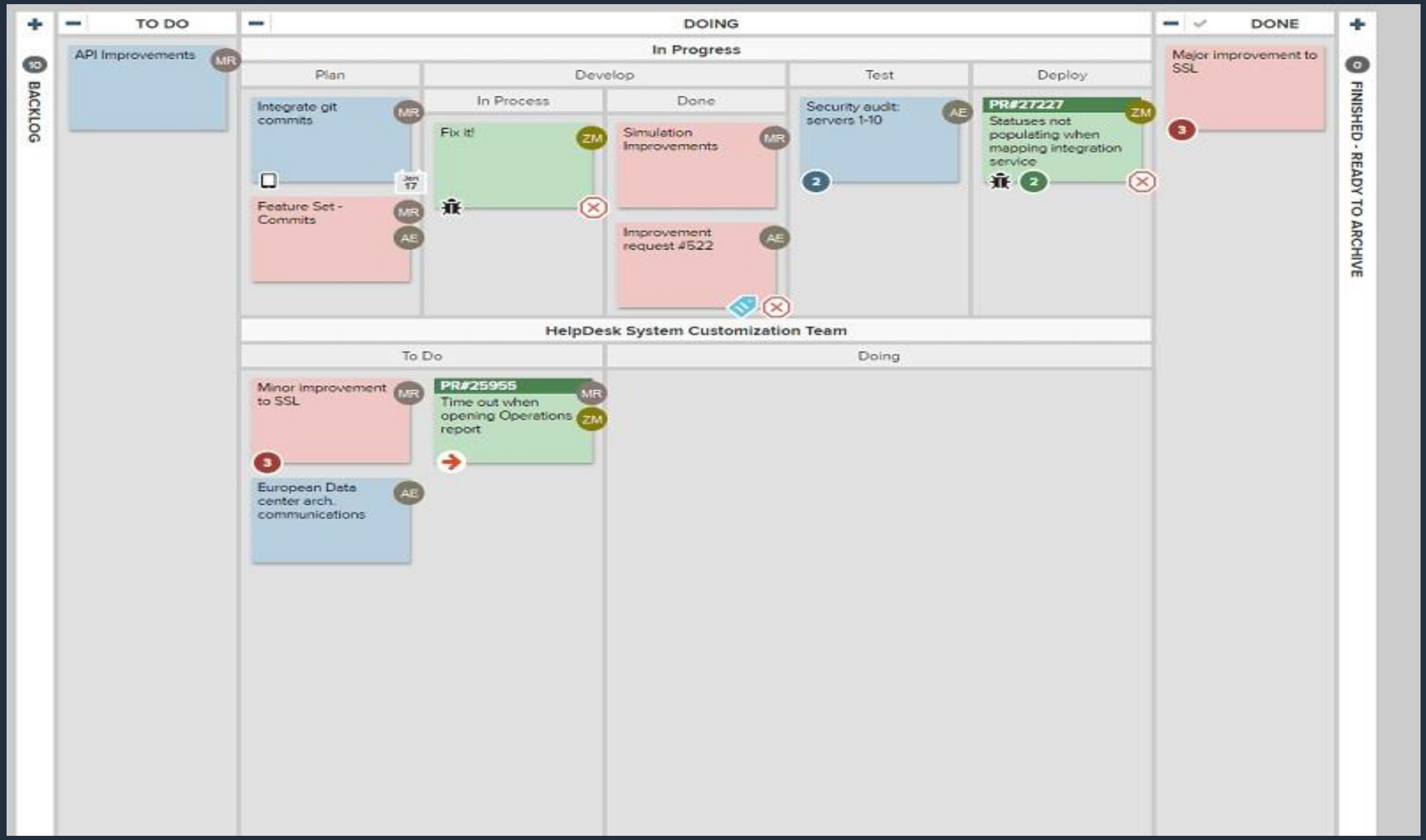
What and Why – KANBAN?



The COMMITMENT POINT is the moment when an idea is picked up by the team and work starts on the project.

The DELIVERY POINT is the end of a kanban team's workflow.

KANBAN Board Example



Scrum vs Kanban

	Scrum	Kanban
Origin	Software development	Lean manufacturing
Idea logy	Learn through experiences, self-organize and prioritize, and reflect on wins and losses to continuously improve.	Use visuals to improve work-in-progress
Cadence	Regular, fixed-length sprints (i.e. two weeks)	Continuous flow
Practices	Sprint planning, sprint, daily scrum, sprint review, sprint retrospective	Visualize the flow of work, limit work-in-progress, manage flow, incorporate feedback loops
Roles	Product owner, scrum master, development team	No required roles

The Twelve Factor

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

The Twelve Factor

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes

Architecture Rule of Thumb

- The Separation of Concern
    - We should consider different aspects of a system separately

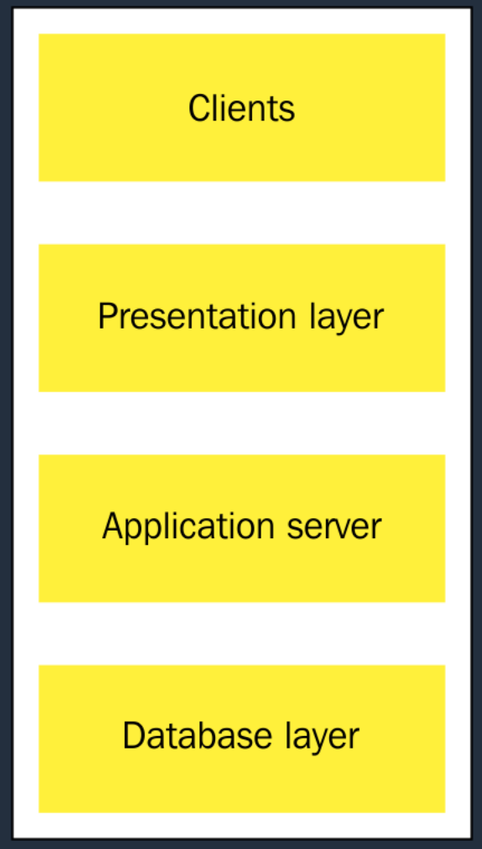
- The Principal of Cohesion
    - Cohesion refers to the degree to which the elements of a software module belong together. It is desirable to have strong cohesion in a module.

- Coupling
    - Coupling refers to the degree of dependency between two modules. We always want low coupling between modules.

Systems with high cohesion and low coupling would automatically have separation of concerns, and vice versa.

Architecture Example – Three Tier System

- This software architecture pattern has been in use for decades and continues to be popular.
- These types of systems are very common, and it is quite likely that you will encounter them, either as legacy systems or as new designs.



Client Web Browser or Mobile App Clients

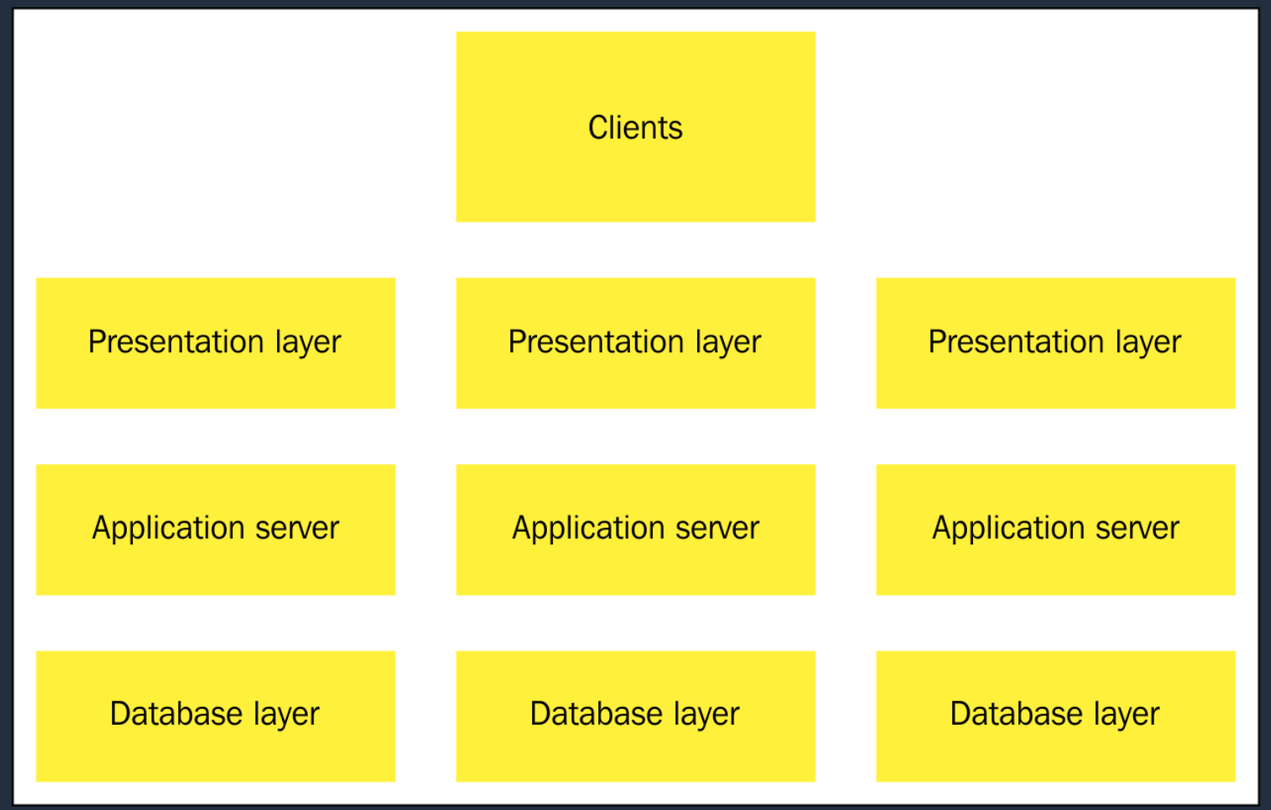
The presentation tier will be a web whose frontend is implemented using the React web framework. It will be deployed as a set of JavaScript and static HTML files.

The logic tier is a backend which is implemented using the Clojure language on the Java platform pr python or Go

In data tier, selected DBMS will be configured and maintained

Architecture Example – Introducing MicroServices

- Microservices is a recent term used to describe systems where the logic tier of the three-tier pattern is composed of several smaller services that communicate with language-agnostic protocols.
- Typically, the language-agnostic protocol is HTTP based, commonly JSON REST, but this is not mandatory. There are several possibilities for the protocol layer.



This architectural design pattern lends itself well to a CD approach since, as we have seen, it's easier to deploy a set of smaller standalone services than a monolith.

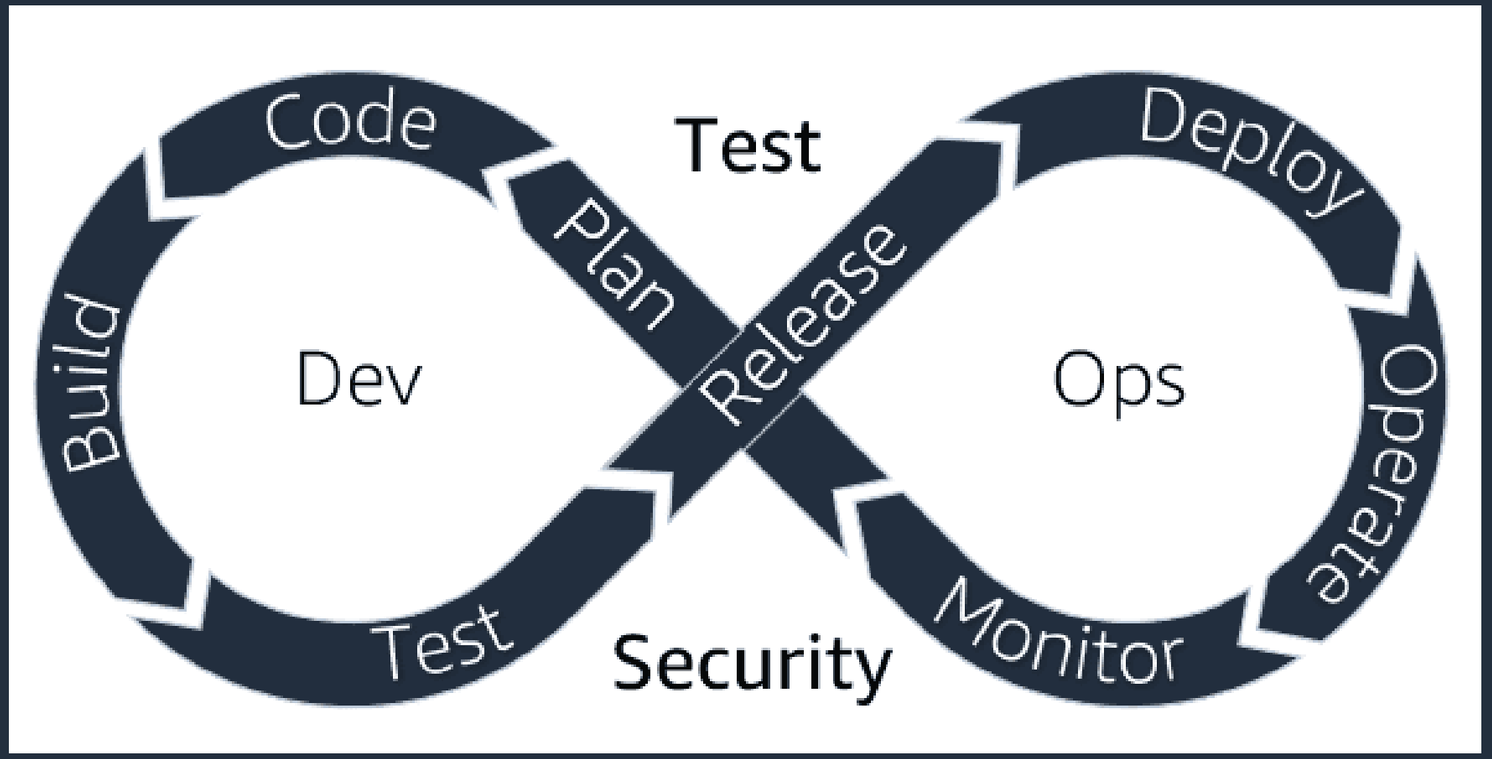
DevOps stages and tools

DevOps – The BUZZ! Word

- DevOps is
  - the combination of cultural philosophies, practices, and tools
  - that increases an organization's ability to deliver applications and services at high velocity:
  - evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.
  - This speed enables organizations to better serve their customers and compete more effectively in the market.
- DevOps emphasizes better collaboration and efficiencies so teams can innovate faster and deliver higher value to businesses and customers.
- DevOps is Combination of
  - Cultural philosophies for removing barriers and sharing end-to-end responsibility
  - Processes developed for speed and quality, that streamline the way people work
  - Tools that align with processes and automate repeatable tasks, making the release process more efficient and the application more reliable

DevOps – Infinity Loop

- DevOps is short for Development (Dev) and Operations (Ops).
- Dev are the people and processes that create software. Ops are the teams and processes that deliver and monitor the software.

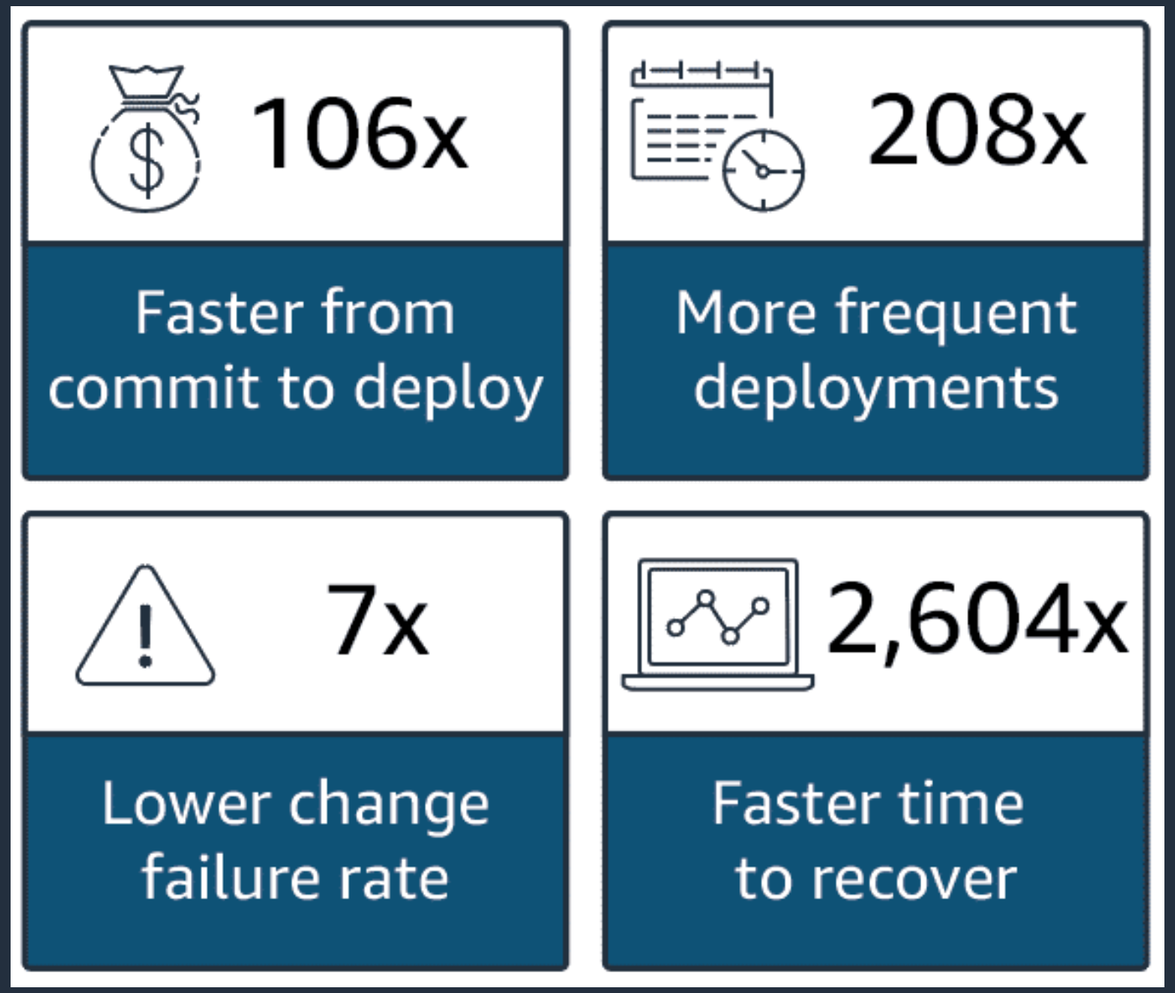


- Developers change things quickly, release often, and measure success by the rate of delivery. Operations are driven by maintaining stability of the application. Frequent releases are a cause for concern of the stability and reliability of the application on the supported platforms, especially during high network traffic.

DevOps brings together formerly siloed roles (development, IT operations, quality engineering, and security) to optimize the productivity of developers and the reliability of operations.

Why DevOps

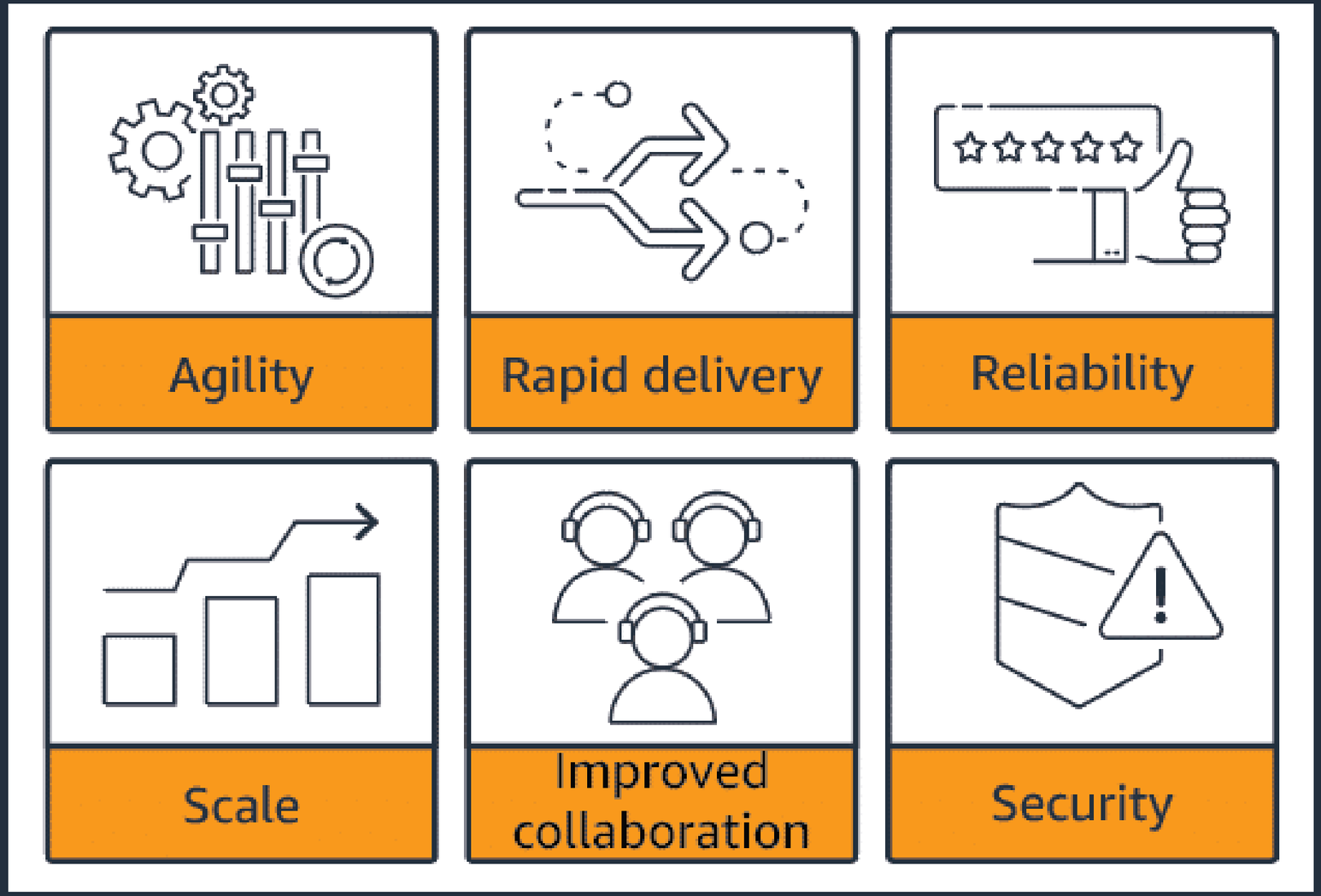
- According to the Accelerate State of DevOps 2019 report, organizations of any size, and in any industry, can benefit from DevOps.



- According to the report, teams who successfully adopt DevOps
  - see shorter delivery cycles,
  - decreased change failure rates, and
  - improved performance.

Why DevOps - Benefits

- Organizations of all sizes, from small startups to big enterprises, can benefit from adopting DevOps. Following are some of the main benefits of DevOps.



Why do some teams initially resist adopting DevOps?

- DevOps will bring change and disrupt the way you work and interact with others
- DevOps will have organizational and team-level impact

- To overcome this reluctance,
    - it is important to understand the value of DevOps,
    - set realistic expectations for the teams.
- To be successful
    - you need buy-in across the organization and development teams.

Knowledge Check

Select the answer that best describes DevOps:

A. DevOps is the combination of development and operations teams, where members of each team must learn to develop, test, and deploy the software.

B. DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity.

C. DevOps is a set of development and operations tools that automate the software development process.

D. DevOps is the coordination between the development and operations teams to improve communication between the teams to develop software more efficiently.

Knowledge Check

Which of the following best describes the challenges associated with traditional software development methods?

A. Traditional software development methods often use older software, outdated development practices, and must be replaced with DevOps software.

B. Traditional software development methods often use a microservices architecture that is difficult to update and deploy, because when changes are made the entire application must be redeployed.

C. Traditional software development methods often use continuous integration and continuous delivery, which slows down the development process, by using repetitive tasks.

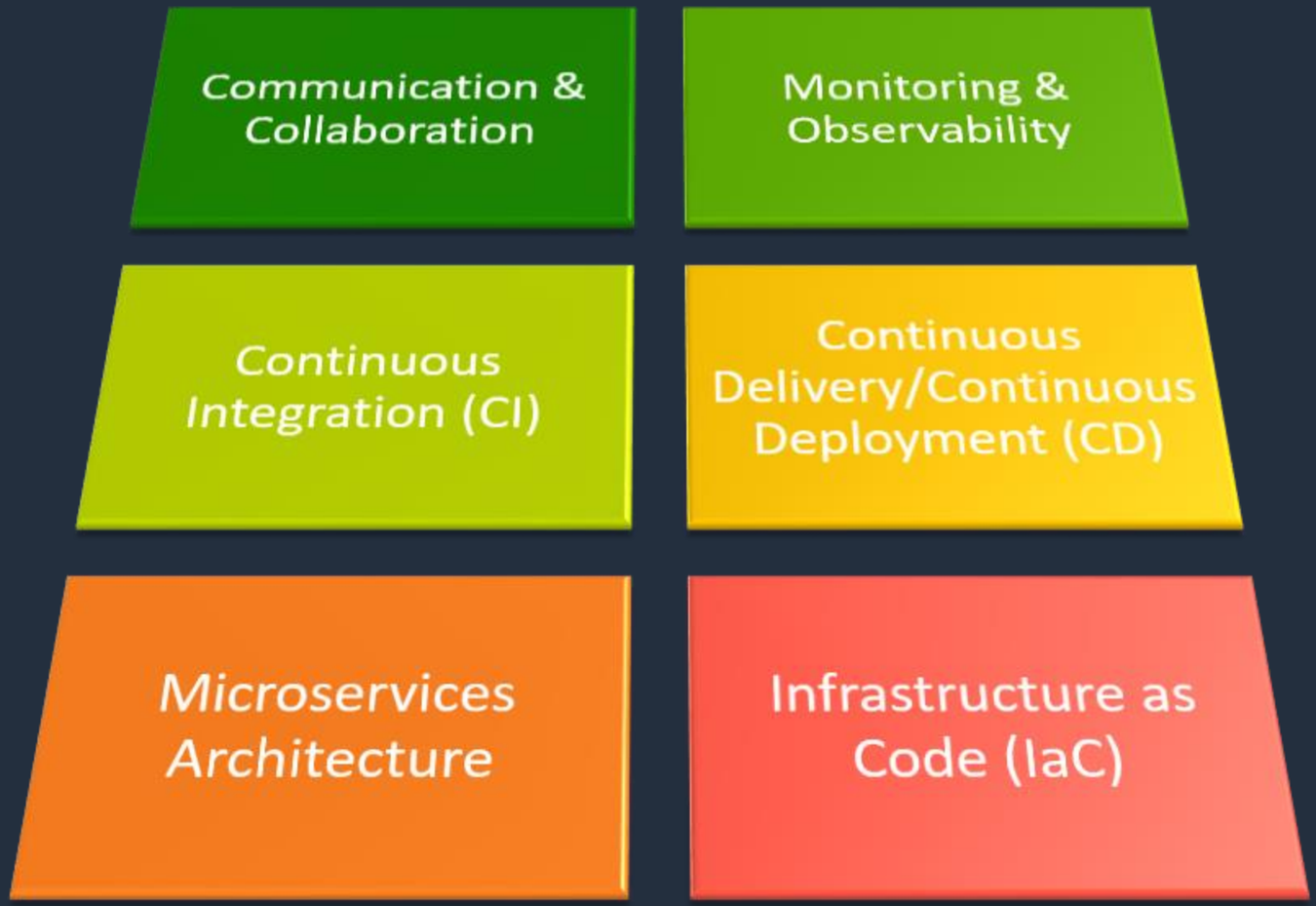
D. Traditional software development methods often involve monolithic architectures that are difficult to update and deploy, development methods that are not iterative, and have long release cycles.

DevOps Culture

- A shift to DevOps requires creating and nurturing a DevOps culture, which is a culture of transparency, effective and seamless collaboration, and common goals.
- There are seven core principles that can help you achieve a DevOps culture:

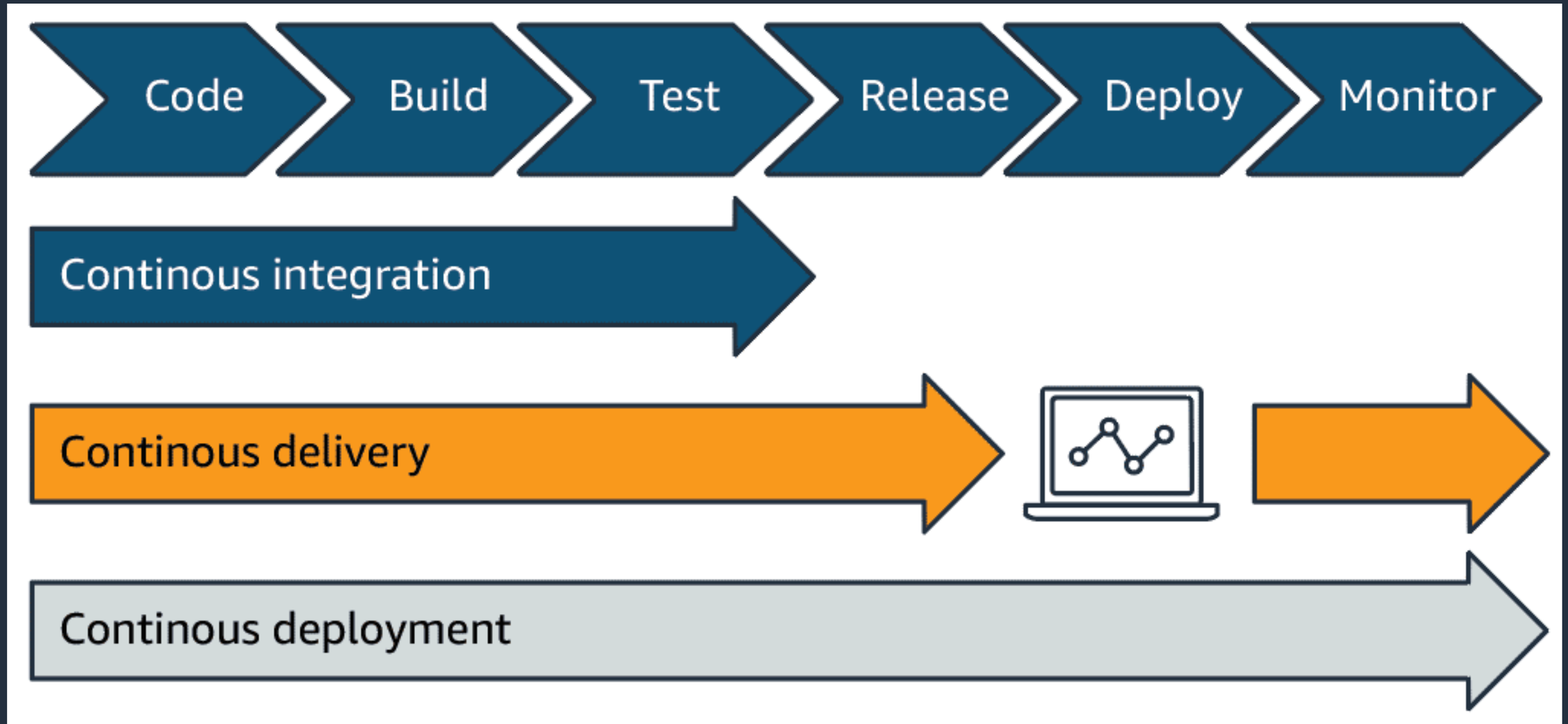
DevOps Practices

- DevOps culture leads to DevOps practices that are geared toward streamlining and improving the development lifecycle, to reliably deliver frequent updates, while maintaining stability.



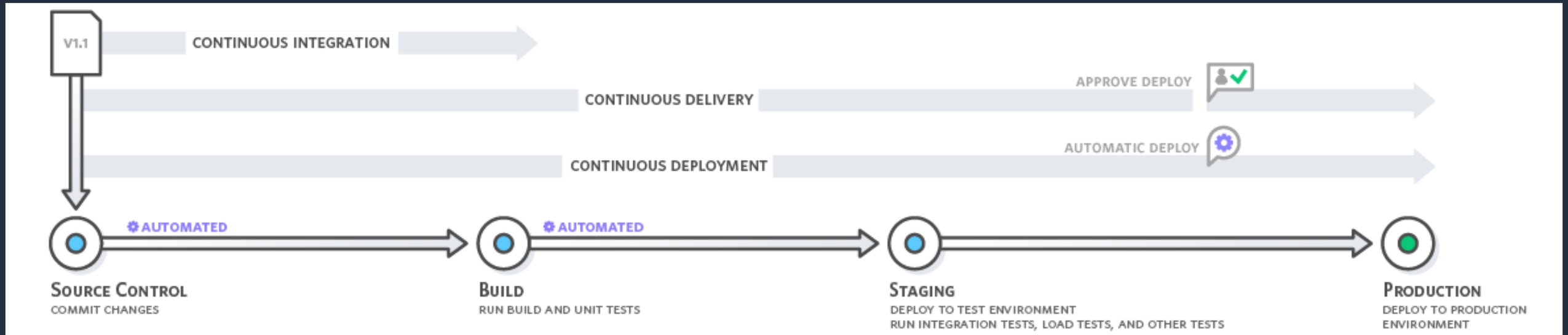
DevOps Pipeline

- A DevOps pipeline is a set of stages that move code from source, all the way to deployment. It depicts typical stages in a DevOps pipeline and depicts the phases in CI/CD pipeline.



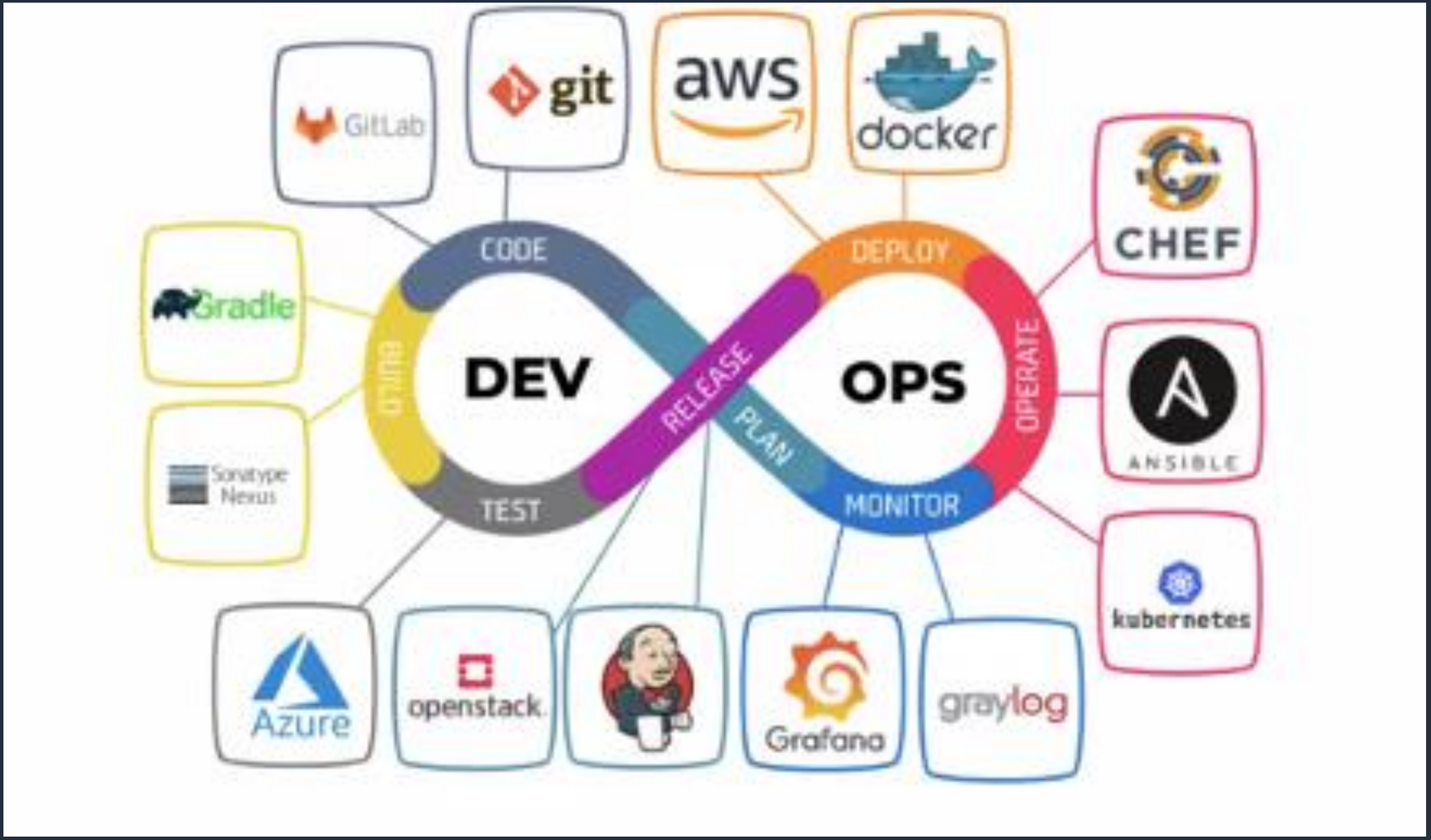
DevOps Pipeline – CI vs CD

- Continuous integration (CI) is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
- Continuous Delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.
- The difference between continuous delivery and continuous deployment is the presence of a manual approval to update to production. With continuous deployment, production happens automatically without explicit approval.



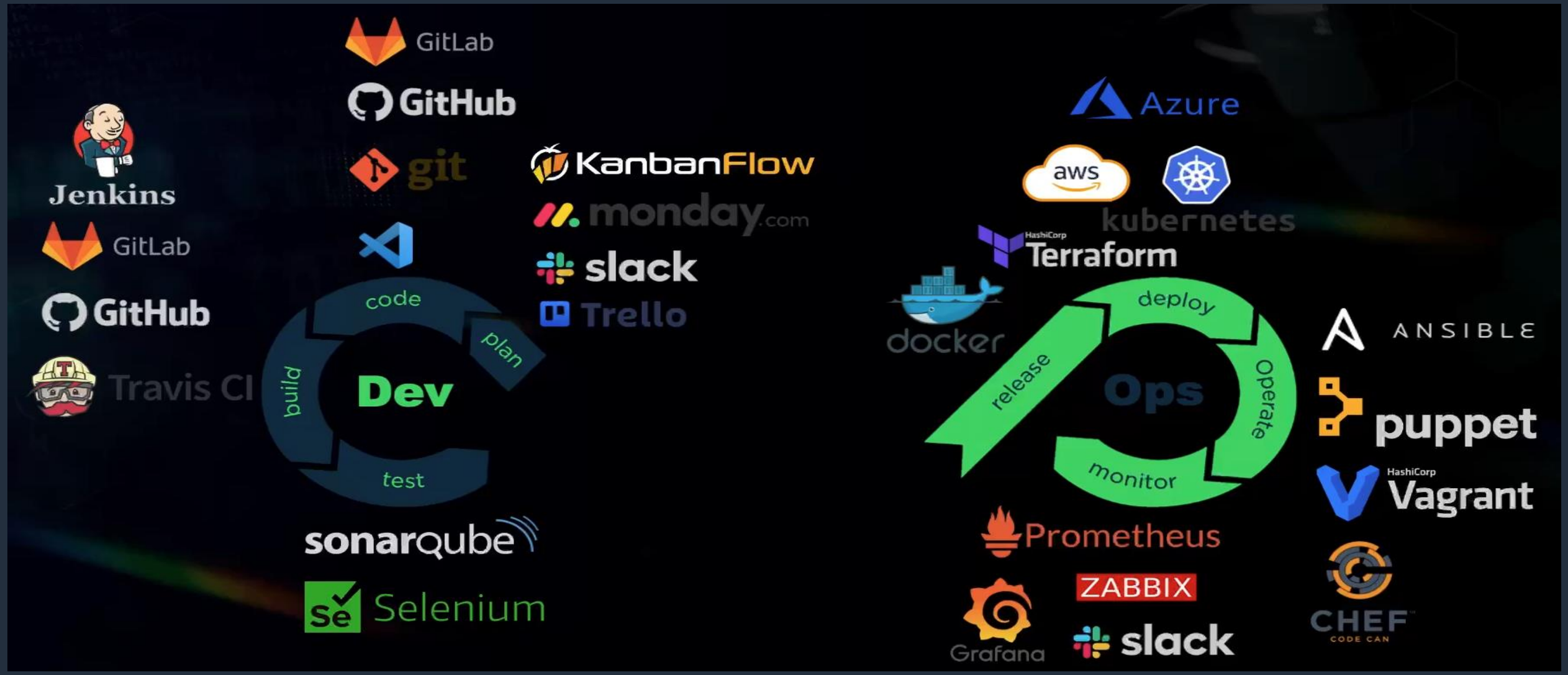
DevOps Tools

- DevOps practices require DevOps tools. Tools make processes easier, consistent, and predictable.



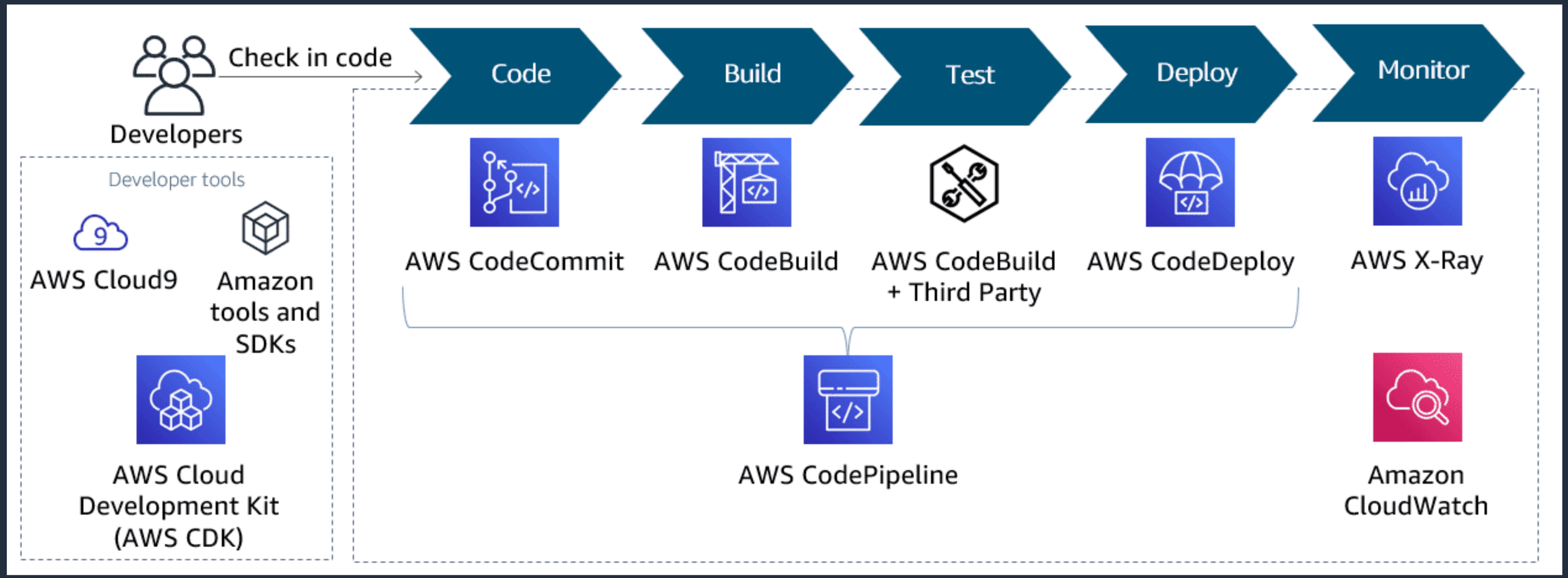
DevOps Tools

DevOps practices require DevOps tools. Tools make processes easier, consistent, and predictable.



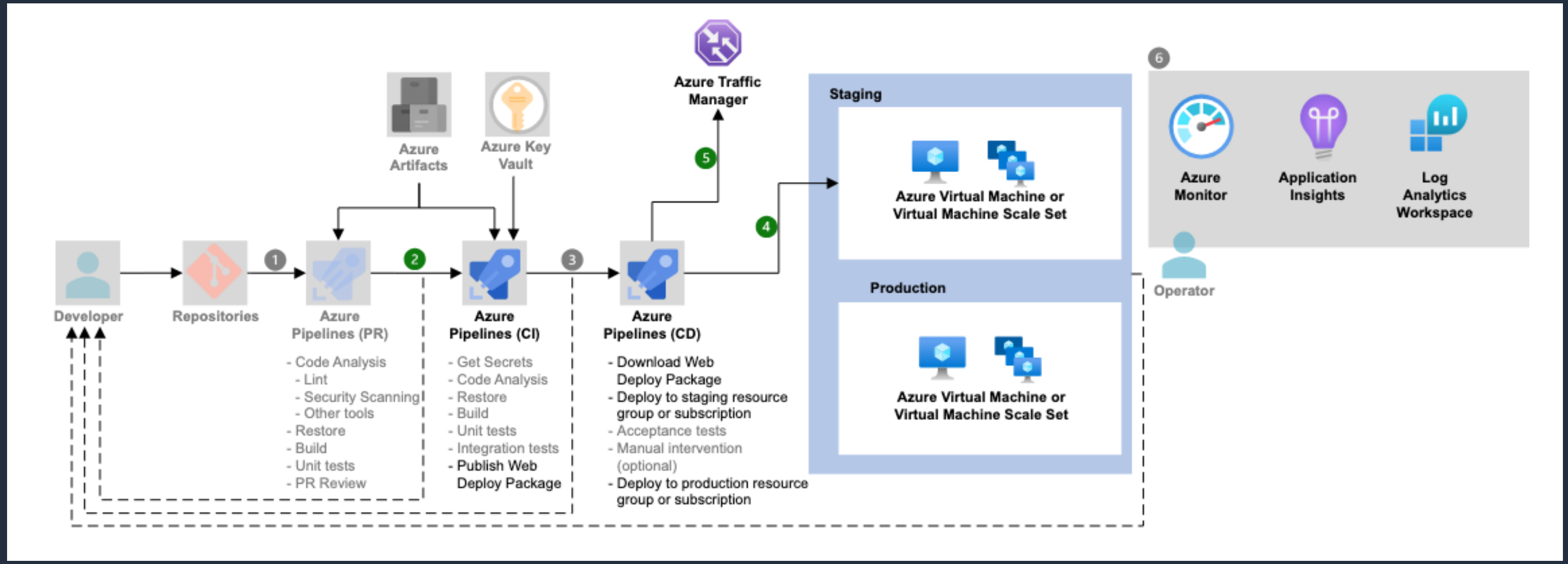
DevOps Tools on AWS

- DevOps practices require DevOps tools. Tools make processes easier, consistent, and predictable.

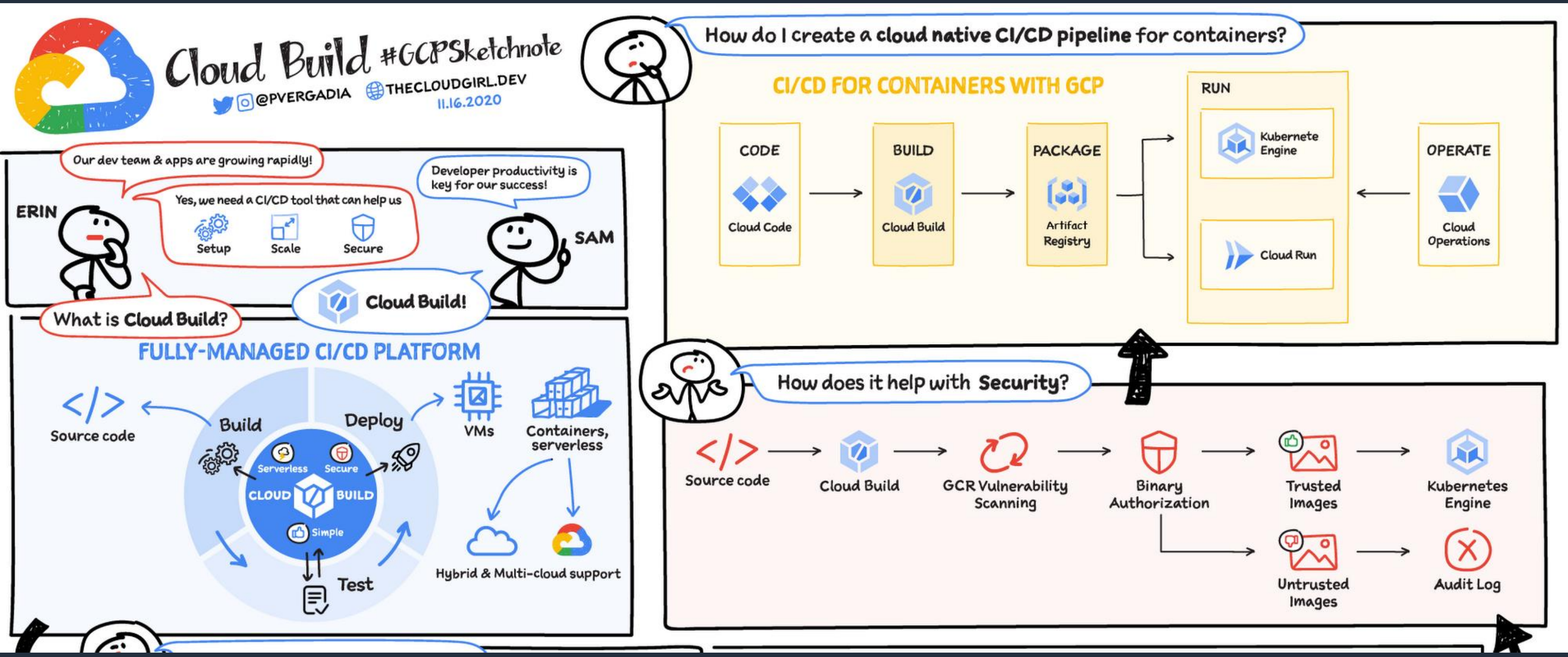


DevOps Tools on Azure

- DevOps practices require DevOps tools. Tools make processes easier, consistent, and predictable.



DevOps Tools on GCP



Knowledge Check

Your management team has decided that they want to move from their current traditional software development model and adopt a DevOps model. What changes will they need to implement to move to DevOps?

A. Moving to DevOps requires hiring DevOps professionals, and investing in new tools.
B. Moving to DevOps requires a cultural shift, implementing DevOps practices, and using the right tools to automate processes.
C. Moving to DevOps requires hiring DevOps professionals, adopting new processes, and investing in new tools.
D. Moving to DevOps requires creating new teams, changing all processes, and selecting the right tools.

The Periodic table of DevOps tools