KHOA CÔNG NGHỆ THÔNG TIN-ĐHKHTN
CSC11004 - MẠNG MÁY TÍNH NÂNG CAO

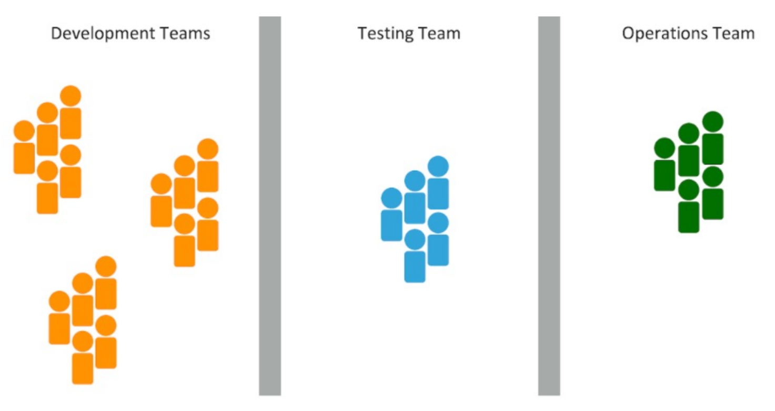
DEVOPS

Lê Ngọc Sơn

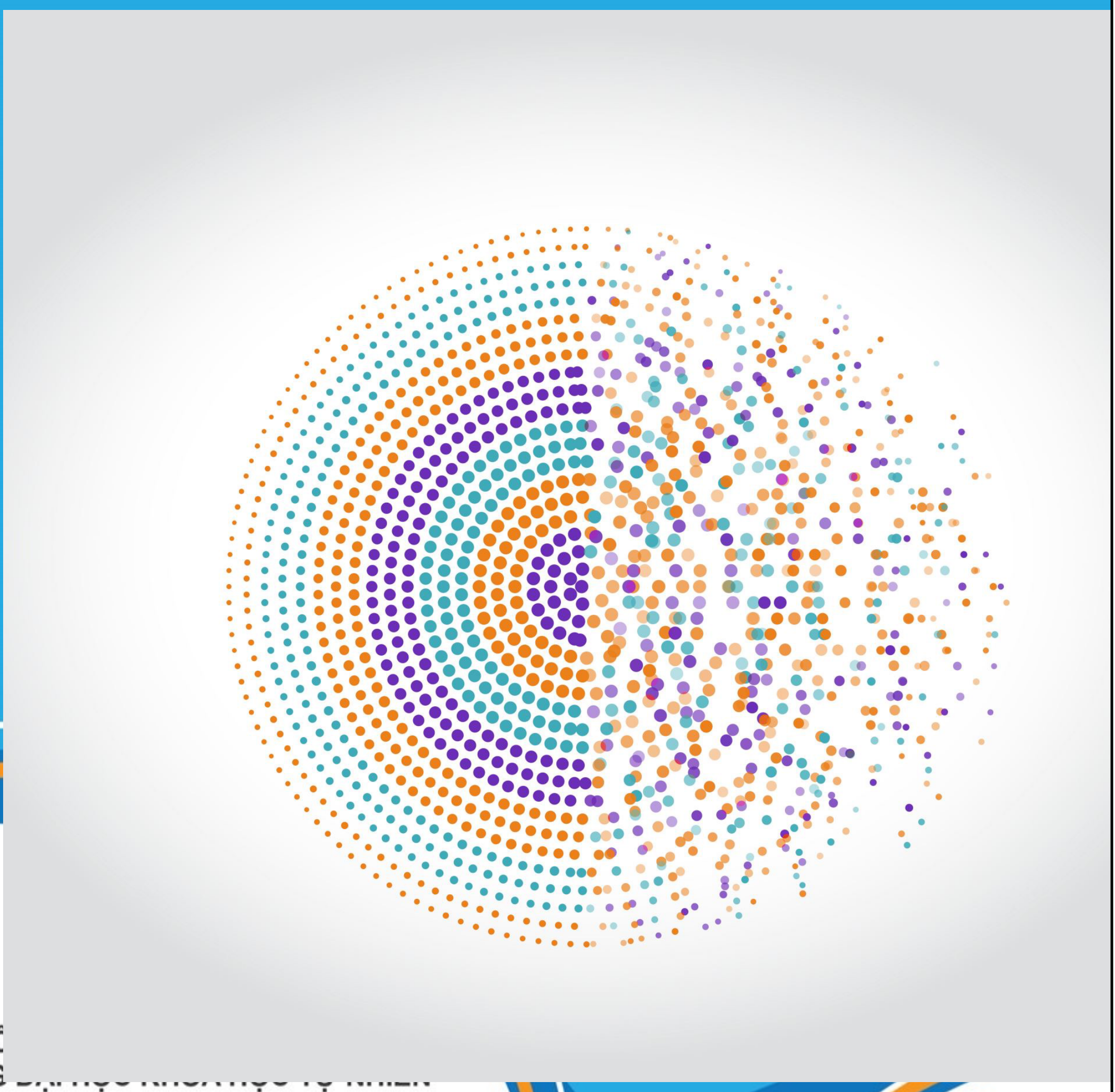
TPHCM, 9-2021

The story of some code: Traditional Silos

- ☐ Devs write code
- ☐ "throw it over the wall" to QA
- ☐ Code bounces back and forth between Dev and QA as QA discover problems and Devs fix them
- ☐ QA/Dev "throws the code over the wall" to Operations
- ☐ Oh no! There's problem. Ops throw it back over the wall to Dev
- ☐ Ops: "Our system are fine; it's your code"
- ☐ Devs: " But the code works on my machine"



The story of DevOps vs Traditional Silos

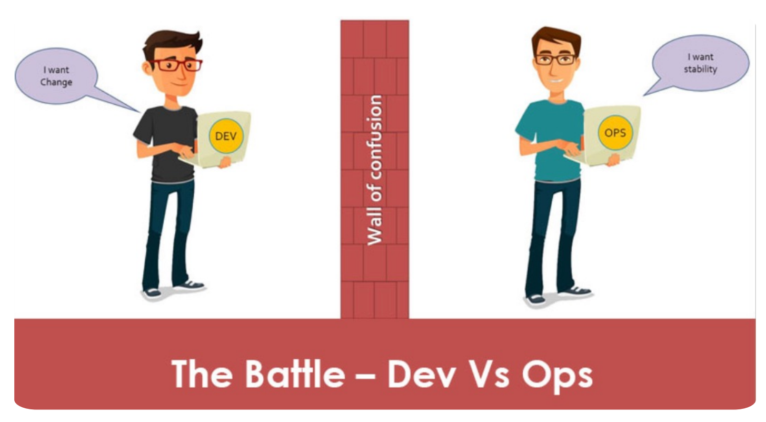


Traditional Silos - What went wrong?

Dev and Ops are black boxes to each other, which leads to finger pointing
- ☐ Because Ops is a black box, Devs don't really trust them
- ☐ And Ops doesn't really trust Devs

Dev and Ops are different priorities, which pits them against each other
- ☐ Ops view Devs as breaking stability
- ☐ Dev see ops as an obstacle to delivering their code

Even if they WANT to work together
- ☐ Dev is measure by delivering features, which means deploying changes
- ☐ Ops is measured by uptime, but changes are bad for stability



Defining DevOps

- ☐ DevOps = **Dev** (Development) + **Ops** (Operation)
- ☐ Different people define DevOps in a variety of ways
- ☐ This definition from Wikipedia is a good starting point:

"DevOps is a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops)

DevOps aims at shorter development cycles,

Downsides of Tradition Silos

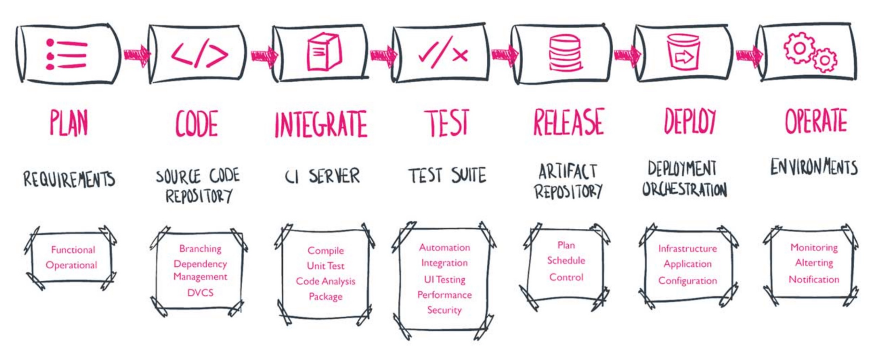
- ☐ "Black boxes" lead to finger pointing
- ☐ Lengthy process means slow time-to-market
- ☐ Lack of automation means things like builds and deployments are inconsistent
- ☐ It takes a long time to identify and fix problems

DevOps is NOT

- ☐ DevOps is NOT tools, but Tools are essential to success in DevOps
- ☐ DevOps is NOT a standard
- ☐ DevOps is NOT a product
- ☐ DevOps is NOT a job title

Story of Some Code: DevOps

- ☐ Devs write code
- ☐ Code commit triggers automated build, integration and tests
- ☐ QA can get their hands on it almost immediately
- ☐ Once it is ready, kick off an automated deployment to production
- ☐ Since everything is automated, it is much easier to deploy while keeping things stable
- ☐ Deployments can occur much more frequently, getting features into the hands of customers faster



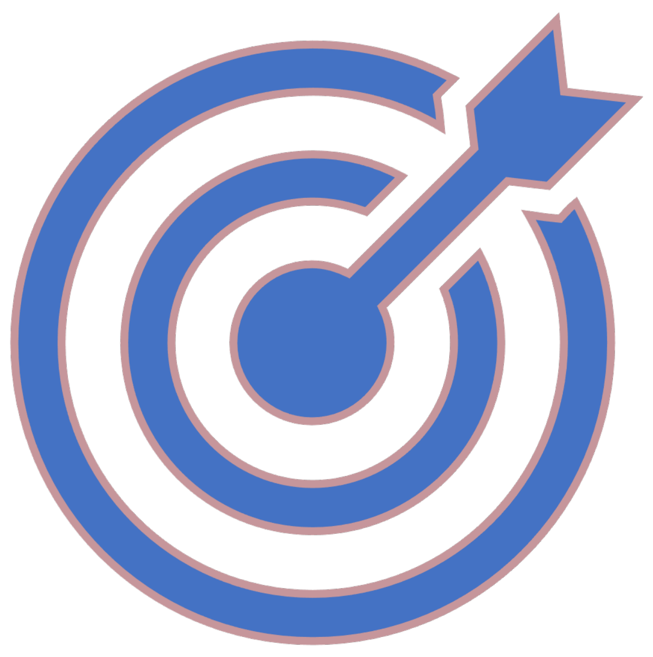
The Goals of DevOps Culture

With DevOps:

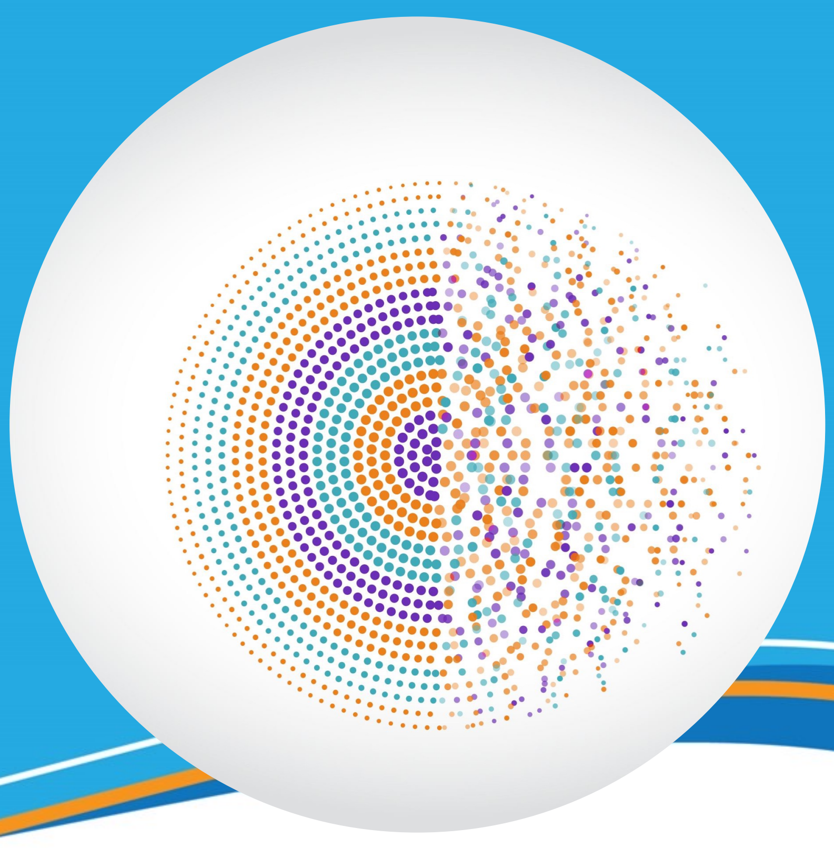
- Dev and Ops are playing on the same team
- Dev and Ops share the same goals

These goals include things like:

- Fast time – to – market
- Few production failures
- Immediate recovery from failures



Build Automation



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

DevOps – What went Right ?

Dev and Ops worked together to build a robust way of changing code quickly and reliably

- Both Dev and Ops worked to prioritize both speed of delivery and stability

Automation led to consistency

- Building, testing and deploying happened the same way every time
- Building, testing and deploying happened much more quickly and more often

Good monitoring, plus the swift deployment process, ensured problems could be fixed even before users noticed them

- Dev and Ops worked together up front to build good processes
- Even though a code change caused a problem, users experienced little or no downtime

Build Automation

- **Build automation** is the process of automating the creation of a software build and the associated processes including: compiling computer source code into binary code, packaging binary code, and running automated tests.
- The tools of build automation often differ depending on what programming languages and frameworks are used, but they have one thing common: automation

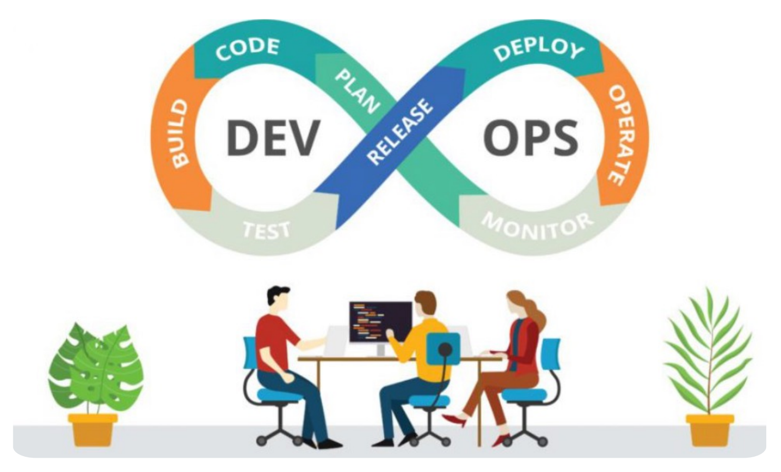
Why do DevOps ?

Happier teams

- Tech employees tend to be happier doing DevOps than under traditional silos
- More time innovating and less time putting out fires
- Devs don't feel like they have to fight to get their work out there
- Ops don't have to fight Dev to keep the system stable

Happier customers

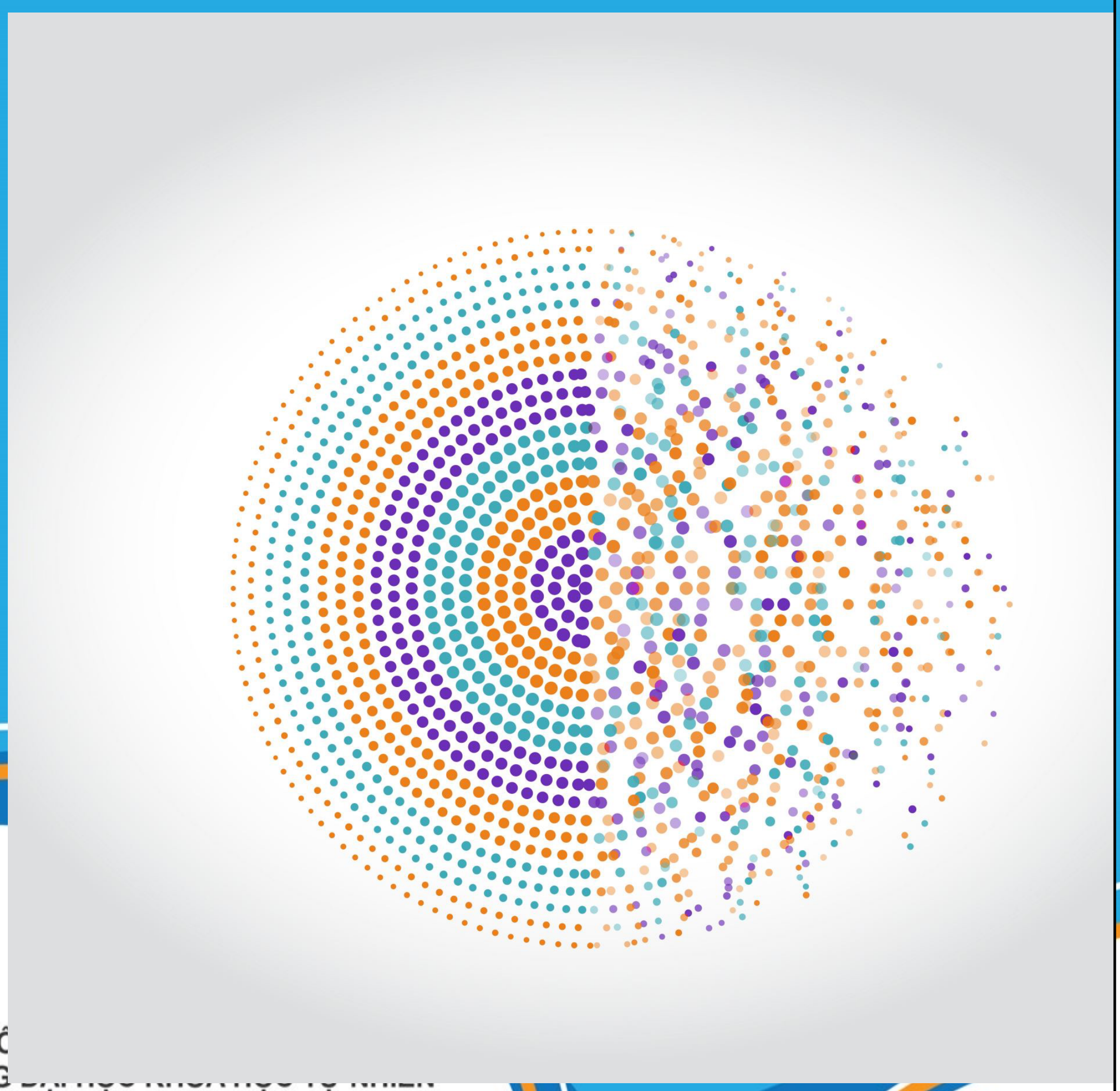
- DevOps lets you give customers the features they want quickly
- And don't have to sacrifice stability to do it



Why do build automation?

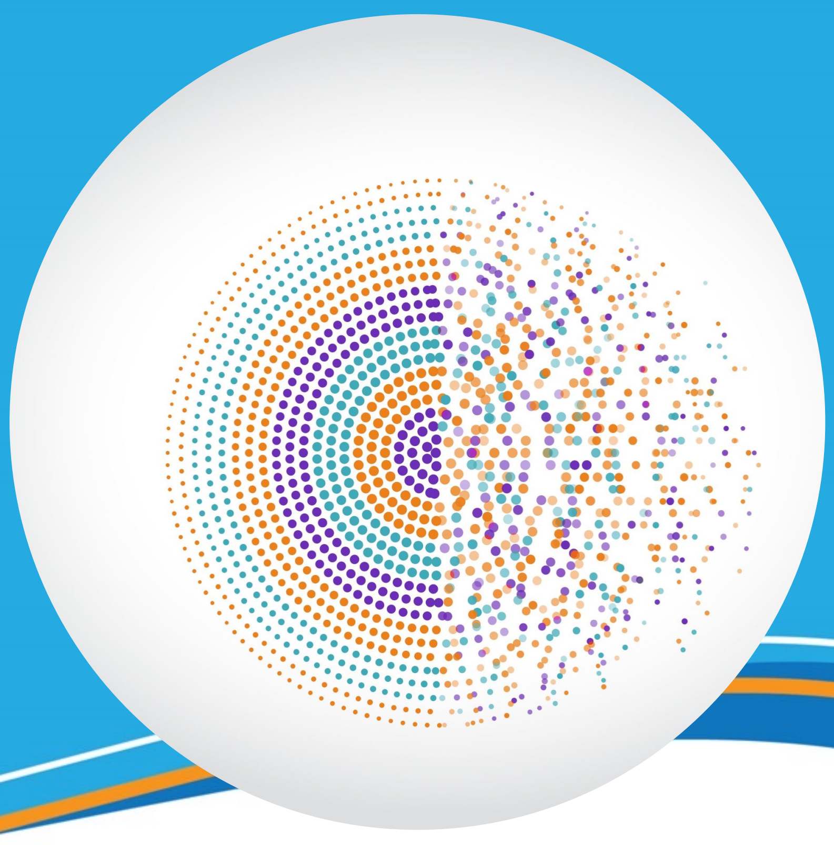
- **Fast** - Automation handle tasks that would otherwise need to be done manually
- **Consistent** – The build happens the same way every time, remove problems and confusion that can happen with manual builds
- **Repeatable** – The build can be done multiple times with same result. Any version of the source code can always be transformed into deployable code in a consistent way
- **Portable** – The build can be done the same way on any machine. Anyone on the team can build their machine, as well as on a shared build server.
- **Reliable** – There will be fewer problems causes by bad builds

DevOps Concepts and Practices



KHOA CÔ
TRƯỜNG

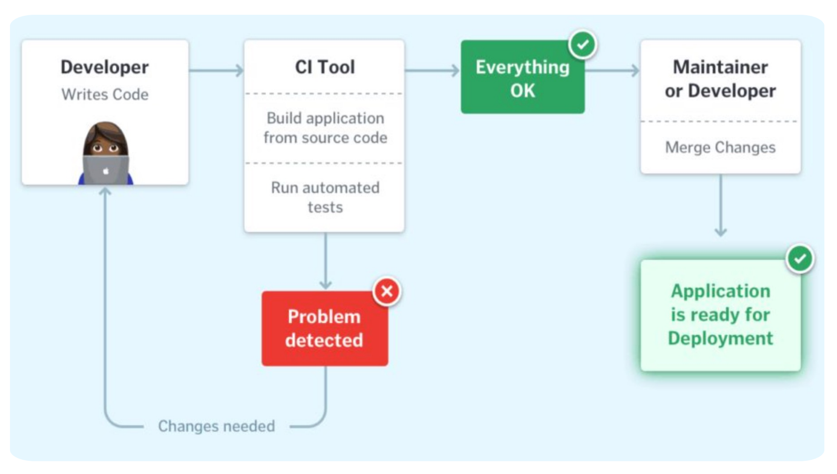
Continuous Integration



KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

What is Continuous Integration ?

- **Continuous Integration (CI):** the practice of frequently merging code changes done by developers
- Traditionally, developers would work separately, perhaps for weeks at a time, and then merge all of their work together at the end in one large effort
- CI means merging constantly throughout the day, usually with the execution of automated tests to detect any problems caused by the merge
- Merging all the time could be a lot of work, so to avoid that it should be automated !



Continuous Deployment

- **Continuous delivery** is a software engineering practice in which code changes are prepared to be released to production. However, keep in mind that the codes must pass the automated unit testing, integration testing, system testing before being pushed to production.
- The transition between continuous integration and continuous delivery is usually completed automatically, including automated testing at the unit, integration, and system levels.
- Automated tests provide more thorough validation. Developers can update and locate issues before the release is publicly available. Additionally, when incorporated with an automated release process, it is a perfect combination to establish a seamless and mechanized pipeline.
- With the releases available in the staging environment, continuous delivery allows developers to release at any rate of their choice, within a single push of a button. That means the decision to initiate a release must be made by a human, and only after that does continuous delivery takes place.

Why do Continuous Integration ?

Early detection – If code doesn't compile or an automated test fails, the developers are notified and can fix it immediately.

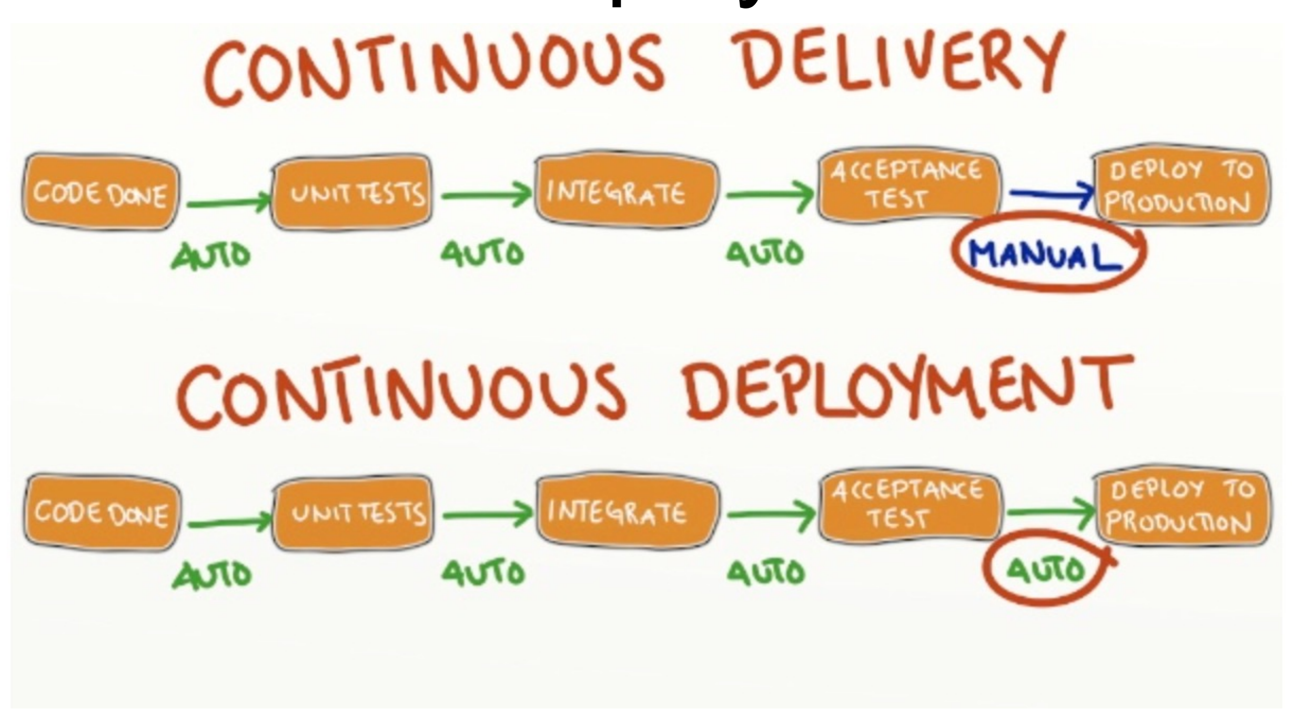
Eliminate the scramble to integrate just before a big release – The code is constantly merged, so there is no need to do a big merge at the end

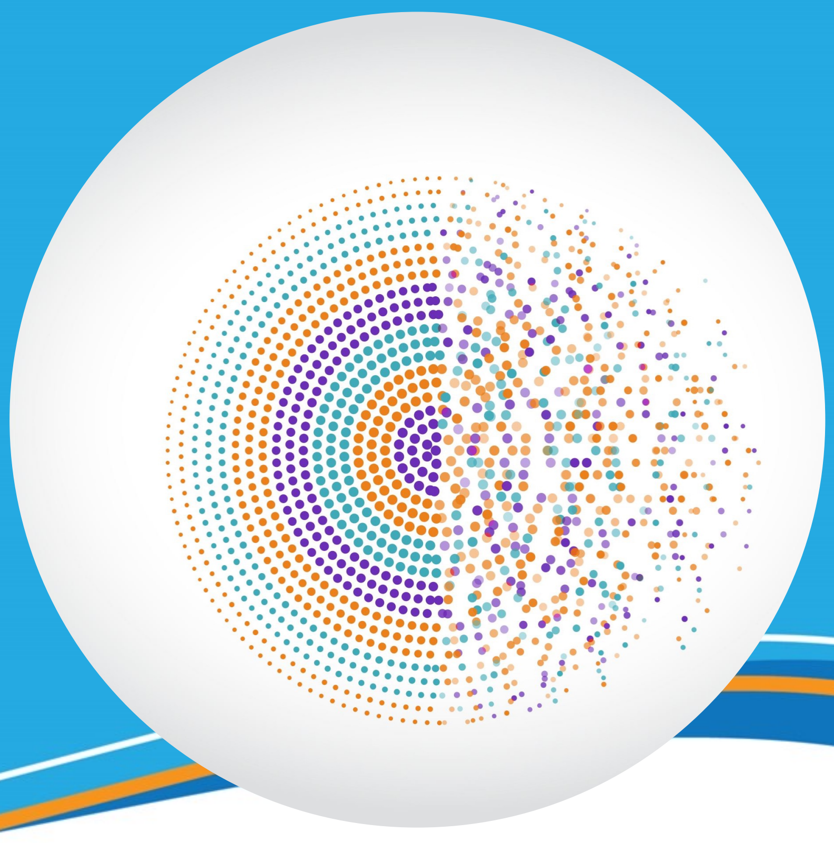
Make frequent release possible – Code is always in a state that can be deployed to production

Makes continuous testing possible – Since the code can always be run, QA testers can get their hands on it all throughout the development process, not just at the end

Encourages good coding practices – Frequent commits encourages simple, modular code

Continuous Delivery vs Continuous Deployment

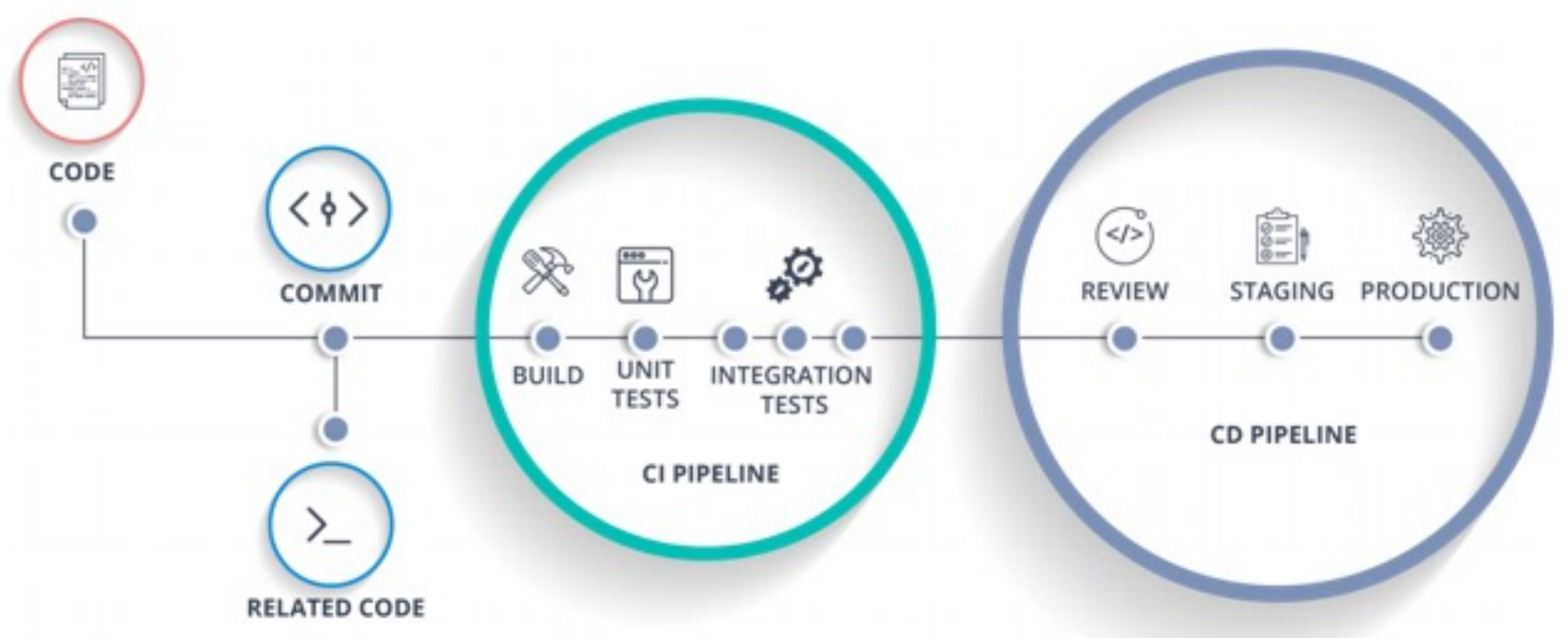




Continuous Delivery

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

CI/CD pipeline



Continuous Delivery

- **Continuous delivery** is a software engineering practice in which code changes are prepared to be released to production. However, keep in mind that the codes must pass the automated unit testing, integration testing, system testing before being pushed to production.
- The transition between **continuous integration** and **continuous delivery** is usually completed automatically, including automated testing at the unit, integration, and system levels.
- Automated tests provide more thorough validation. Developers can update and locate issues before the release is publicly available. Additionally, when incorporated with an automated release process, it is a perfect combination to establish a seamless and mechanized pipeline.
- With the releases available in the staging environment, continuous delivery allows developers to release at any rate of their choice, within a single push of a button. That means the decision to initiate a release must be made by a human, and only after that does continuous delivery takes place.

Infrastructure as Code

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

What is Infrastructure as Code?

- **Infrastructure as Code (IaC):** manage and provision infrastructure through code and automation
- With infrastructure as code, instead of doing things manually, you use automation and code to create an change:
  - Servers
  - Instances
  - Environments
  - Containers

What is Configuration Management ?

- **Configuration Management**: maintaining and changing the state of pieces of infrastructure in a consistent, maintainable and stable way
- Changes always need to happen – configuration management is about doing them in maintainable way
- Configuration management allows you to minimize configuration drift – the small changes that accumulate overtime and make systems different from one another and harder to manage
- Infrastructure as Code is very beneficial for configuration management

What does infrastructure as code look like?

Without infrastructure as code you might:
- ssh into a host
- issue a series of commands to perform the change

With infrastructure as code:
- Change some code or configuration file that can be used with an automation tool to perform changes
- Commit them to source control
- Use an automation tool to enact the changes defined in the code and/or configuration files

With IaC, provisioning new resources and changing existing resources are both done through automation

Configuration Management Example

- You need to upgrade a software package on a bunch of servers:
  - Without good configuration management, you have to log into each server and perform the upgrade. However, this can lead to a lot of problems.
  - With good configuration management, you define the new version of software package in a configuration file or tool and automatically roll out the change to all of the servers.

Why do infrastructure as code ?

- **Consistency in creation and management of resources** – The same automation will run the same way every time
- **Reusability** - Code can be used to make the same change consistently across multiple hosts and can be used again in the future
- **Scalability** – Need a new instance? You can have one configured exactly the same way as the existing instances in minutes (or second)
- **Self- documenting** – With IaC, changes to infrastructure document themselves to a degree. The way a server is configured can be viewed in source control, rather than being a matter of who logged in to the server and did something
- **Simplify the complexity** – Complex infrastructures can be stood up quickly once they are defined as code. A group of several interdependent servers can be provisioned on demand

Why do configuration management?

- **Save time** – It takes less time to change the configuration
- **Insight** – With good configuration management, you can know about the state of all pieces of a large and complex infrastructure
- **Maintainability** – A more maintainable infrastructure is easier to change in a stable way
- **Less configuration drift** – It is easier to keep a standard configuration across a multitude of hosts.

Configuration Management

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Orchestration

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

What is Orchestration ?

- **Orchestration**: automation that supports processes and workflows, such as provisioning resources
- With orchestration, managing a complex infrastructure is less like being a builder and more like conducting an orchestra.
- Instead of going out and creating a piece of infrastructure, the conductor simply signals what needs to be done and the orchestra performs it:
  - The conductor does not need to control every detail
  - The musicians (automation) are able to perform their piece with only a little bit of guidance.

What is monitoring ?

- **Monitoring**: The collection and presentation of data about the performance and stability of services and infrastructure
- Monitoring tools collect data over things such as:
  - Usage of memory
  - Cpu
  - Disk i/o
  - Other resources overtime
  - Application logs
  - Network traffic
- The collection data is presented in various form, such as charts and graphs, or in the form of real-time notifications about the problem

What does Orchestration look like ?

Here is an example:

- A customer requests more resources for a web service that is about to see a heavy increase in usage due to a planned marketing effort.
- Instead of manually standing up new nodes, operation engineers use an orchestration tool to request five more nodes to support the service
- A few minutes later, the tool has five new nodes up and running

A much cooler example:

- A monitoring tool detects an increased load on the service
- An orchestration tool responds to this by spinning up additional resources to handle the load
- When the load decreases again, the tool spins the additional resources back down, freeing them up be used by something else
- All of this happens while the engineer is getting coffee.

What does monitoring look like ?

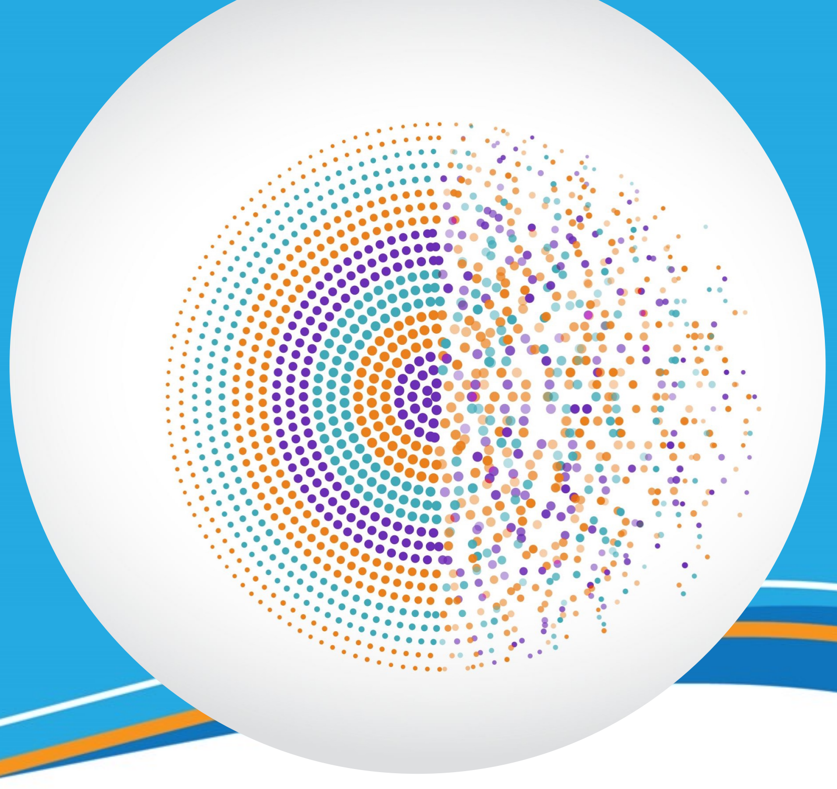
- **Real-time notifications:**
  - Performance on the website is beginning to slow down
  - A monitoring tool detects that response time are growing
  - An administrator is immediately notified and is able to intervene before downtime occurs
- **Postmortem analysis:**
  - Something went wrong in production last night
  - It's working now, but we don't know what caused it
  - Luckily, monitoring tools collected a lot of data during the outage
  - With that data, developers and operation engineers are able to determine the root cause (a poorly performing SQL query) and fix it

Why do orchestration?

- **Scalability** – Resources can be quickly increased or decreased to meet changing needs.
- **Stability** – Automation tools can automatically respond to fix problems before users see them
- **Save time** – Certain tasks and workflow can be automated, freeing up engineer's time
- **Self-service** – Orchestration can be used to offer resources to customers in a self-

Why do monitoring?

- **Fast recovery** – The sooner a problem is detected, the sooner it can be fixed. You want to know the problem before customer does.
- **Better root cause analysis** – The more data you have, the easier it is to determine the root cause of a problem
- **Visibility across the team** – Give useful data to both developers and operations people about the performance of code in production
- **Automated response** – Monitoring data can be used alongside orchestration to provide automated responses to events, such as automated recovery from failures.

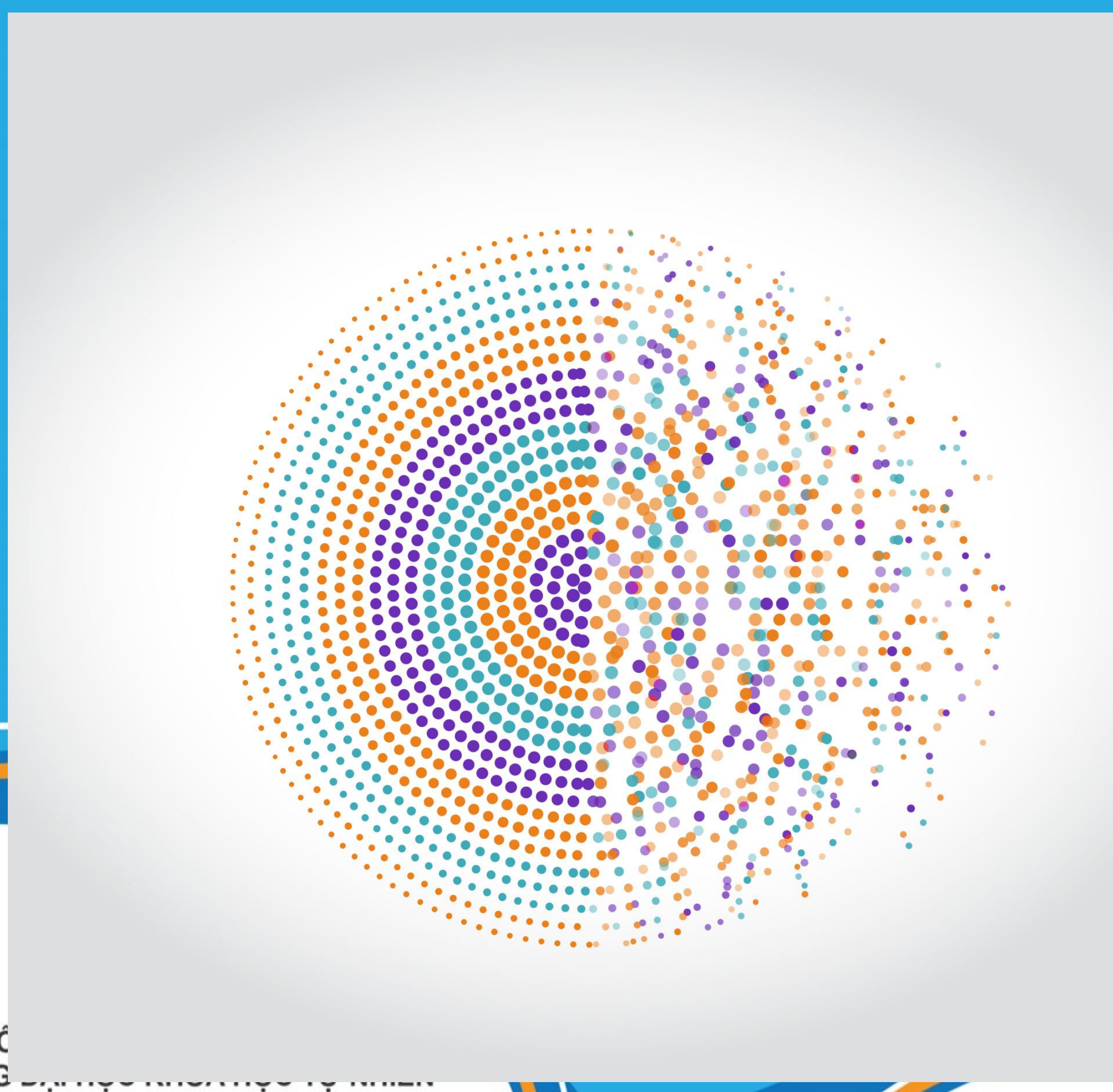


Monitoring

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

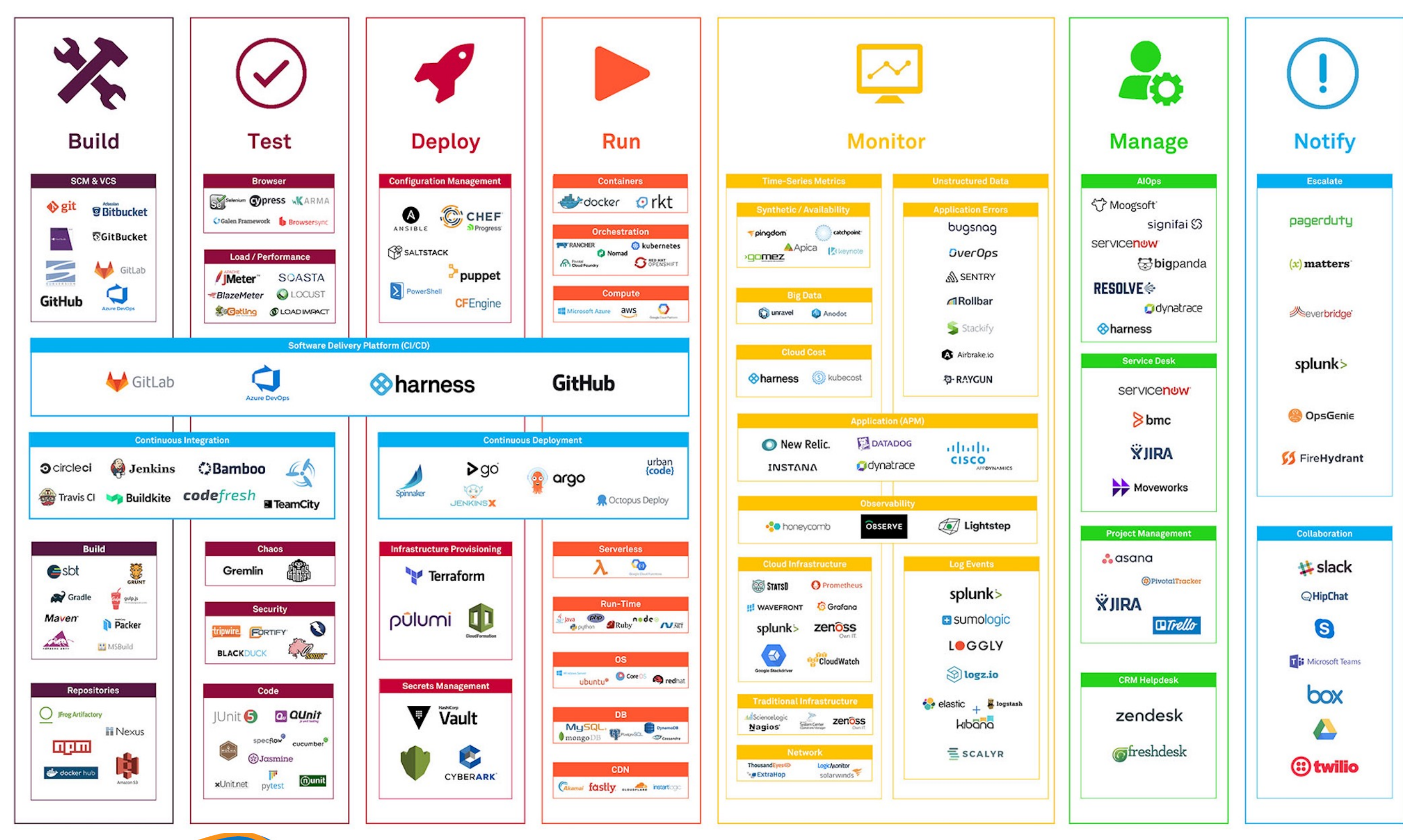
DevOps Tools



KHOA CÔ
TRƯỜNG

fit@hcmus

DevOps Tools Ecosystem 2021



Q&A