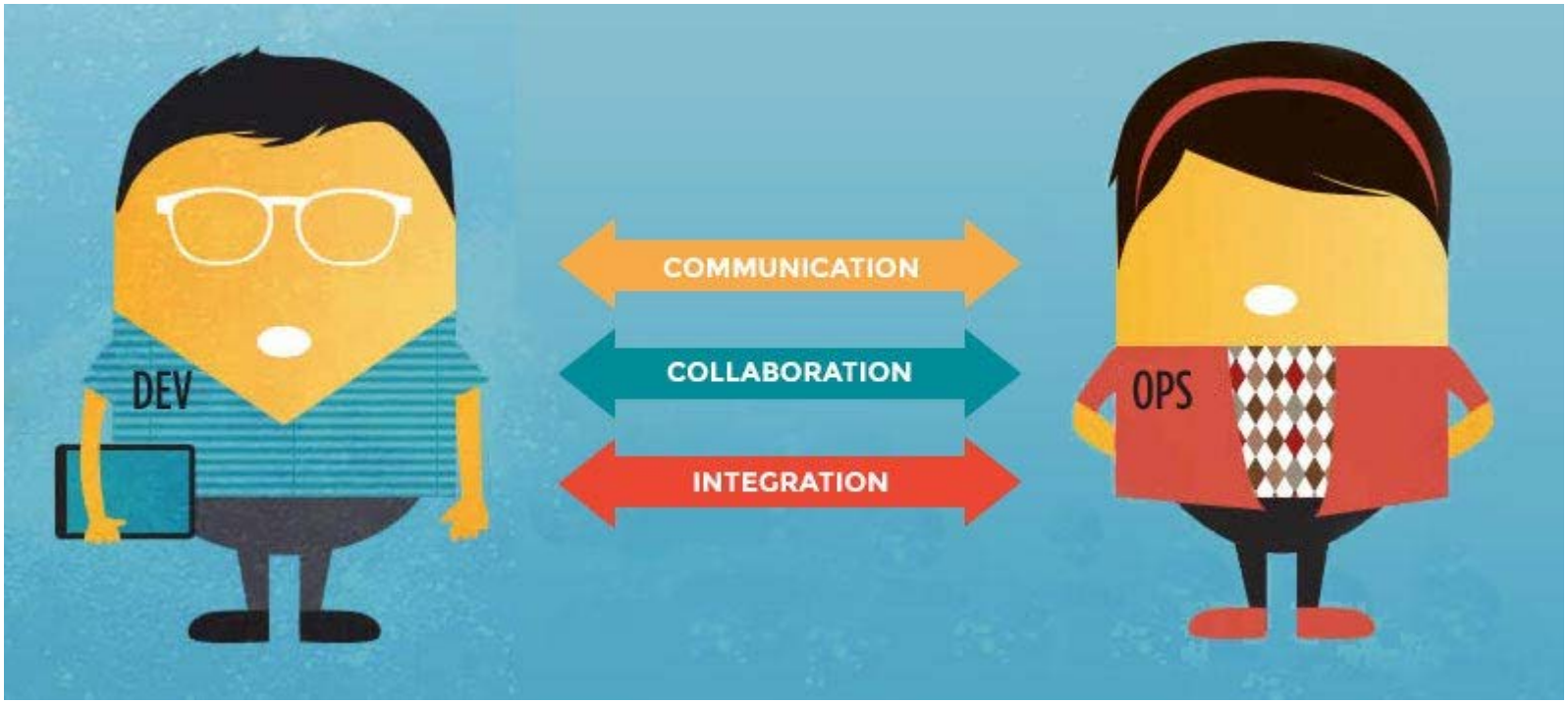

introducing
DevOps


What is DevOps?

What is DevOps?

What Is DevOps ?



What is DevOps?

- DevOps (a combination of development and operations) is a software development method that stresses communication, collaboration and integration between software developers and information technology(IT) professionals thereby
  - Enable rapid evolution of products or services
  - Reduce risk, improve quality across portfolio, and reduce costs

- **DevOps** (a clipped compound of development and operations) is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

- It aims at establishing a culture and environment where building, testing, and releasing software can happen rapidly, frequently, and more reliably.

- DevOps integration targets product delivery, quality testing, feature development and maintenance releases in order to improve reliability and security and faster development and deployment cycles.

- The adoption of DevOps is being driven by factors such as:

- Use of agile and other development processes and methodologies

- Demand for an increased rate of production releases from application and business stakeholders

- Increased usage of data center automation and configuration management tools.

Roles of Devs and Ops

Devs

- Create Change
- Add Or Modify Features

Ops

- Create Stability
- Create or Enhance

DevOps Principles

- Develop and test in an environment similar to production
- Deploy builds frequently
- Validate operation quality continuously

DevOps Life Cycle

- It can be look like this and It can be summed up with the acronym C.A.M.S.

- C – Culture
- A – Automation
- M – Measurement
- S – Sharing

- Culture

- DevOps needs a change in attitude so shared ownership and collaboration are the common working practices in building and managing a service.

- This culture change is especially important for established organizations.

- Automation

- Many business processes are ready to be automated.

- Automation removes manual, error-prone tasks – allowing people to concentrate on the quality of the service.

- Common areas that benefit from automation are:

- Release management (releasing software)
- Configuration management
- Systems integration
- Monitoring
- Testing

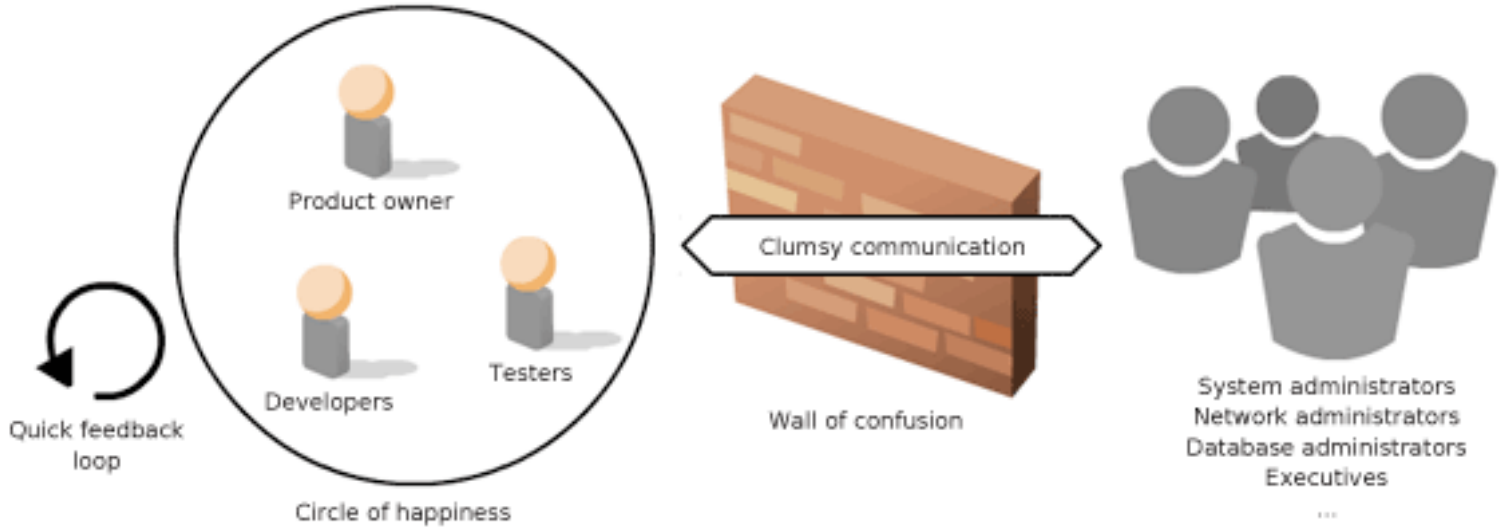
- Measurement

- Data can be incredibly powerful for implementing change, especially when it's used to get people from different groups involved in the quality of the end-to-end service delivery.

- Data is collected on everything and there are mechanisms in place that provide visibility into all systems.

- Sharing
- People from different backgrounds often have different, but overlapping skill sets.
- Sharing between groups will spread an understanding of the different areas behind a successful service, so encourage it.
- Resolving issues will then be more about working together and not negotiating contracts.

DevOps, bridging development and operations



- The development team (developers, testers and product owners) sits inside this circle with regular feedback loops between iterations and smooth communication.

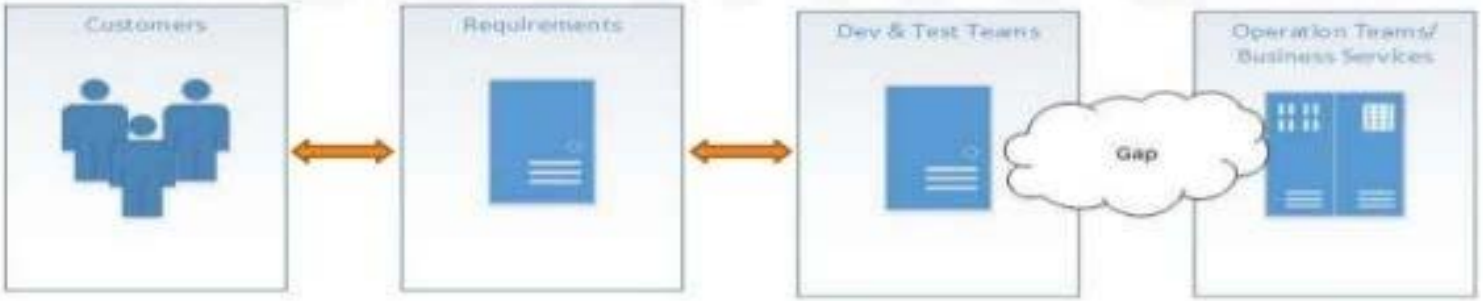
- But communication with people outside this circle is clumsy and this brings a lot of problems that affects the delivery of value.

Why DevOps ?

The benefits of a DevOps approach are many, including:

- Improved deploy frequency which can lead to faster time to market
- Lower failure rate
- Shortened lead time (the amount of **time** it takes to deliver products to the market)
- Faster mean time to recovery

Why DevOps? – Delivery Challenges



why Gaps?

- Dev View:
- Mostly deliver features after testing in development systems.
- Dev systems may not be same as production system
- Developers will have faster turn around time w.r.t features
- Not much concerned about the infrastructural as well as deployment impact because of the code change

Why Gaps?

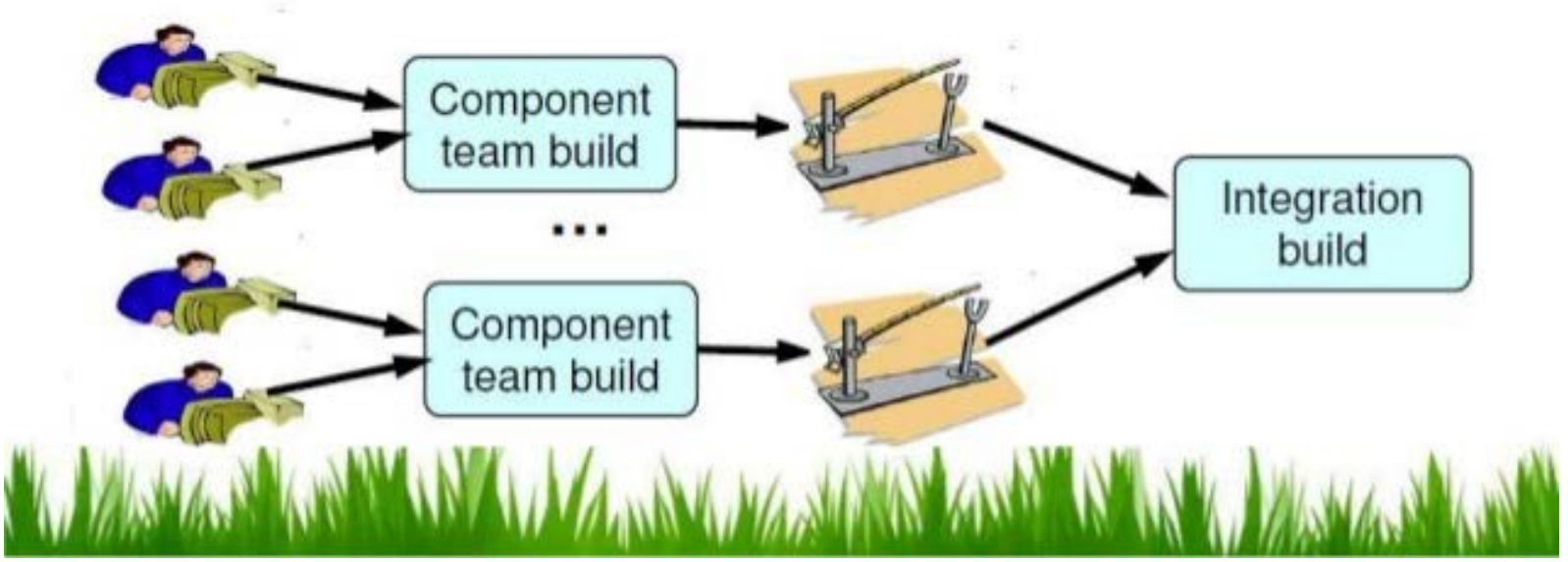
- Ops View:
- Worries more about PSR
- Rewarded mainly for uptime
- Lesser turnaround time w.r.t feature deployment and testing due to large number of dev builds coming their way
- Very much concerned about the infrastructural as well as deployment impact because of the code changes.

Devs and Ops

- Developers work with Ops to understand the impact of code changes
- Developers now work more closely with production-equivalent systems
- Developers focuses on metrics required by Ops team like PSR
- Closely monitors the Dev-Test-Prod pipeline for each deployment and immediate feedback
- Better collaboration and communication

Continuous Integration

- Integrate the code changes by each developer so that the main branch remains up-to-date



- Continuous Integration is a key component of agile practice that ensures software is built and tested regularly and release bug-fixes rapidly.

- Software engineering practice in which changes are immediately tested and reported on when they are added to a larger code base.

- It requires members of a team to integrate their work frequently on a daily basis.

- Each check-in is then verified by an automated build, which allows early detection of defects.

- Goal of CI is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible.

- Since Continuous Integration identifies defects early in the development, defects are usually less complex and easy to resolve.

- Employing Continuous Integration tools and automation testing tools is typical in a DevOps cycle.

- Continuous integration relies on the following principles.
- **Maintain a code repository** (a central location in which data is stored and managed)
- **Automate the build**
- **Make the build self-testing**
- **Keep the build fast**
- **Make it easy to get the latest deliverables**
- **Everyone can see the results of the latest build**
- **Automate deployment**

Continuous Delivery

- Continuous Delivery is the concept that takes Continuous Integration to the next level.
- **“Continuous Delivery is a small build cycle with short sprints…”**
- Where the aim is to *keep the code in a deployable state at any given time*.
- This does not mean the code or project is 100% complete, but the feature sets that are available are examined, tested, debugged and ready to deploy, although you may not deploy at that moment.

Continuous Testing

- **Continuous testing** is the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with a software release.

- Continuous Testing plays a crucial role in Continuous Delivery.

- It implicates usage of methods and concepts of agile development for the testing and QA process, offering more efficient testing process.

- Scope of Continuous Testing:

- Continuous testing includes the validation of both functional requirements and non-functional requirements.

- For testing functional requirements (functional testing), Continuous Testing often involves unit tests, API testing, integration testing, and system testing.

- For testing non-functional requirements (non-functional testing - to determine if the application meets expectations around performance, security etc.), it involves practices such as static code analysis, security testing, performance testing, etc.

- Tests should be designed to provide the earliest possible detection (or prevention) of the risks that are most critical for the business or organization that is releasing the software.

- **Goals and benefits**

- The goal of continuous testing is to provide fast and continuous feedback regarding the level of business risk in the latest build or release.

- This information can then be used to determine if the software is ready to progress through the delivery pipeline at any given time.

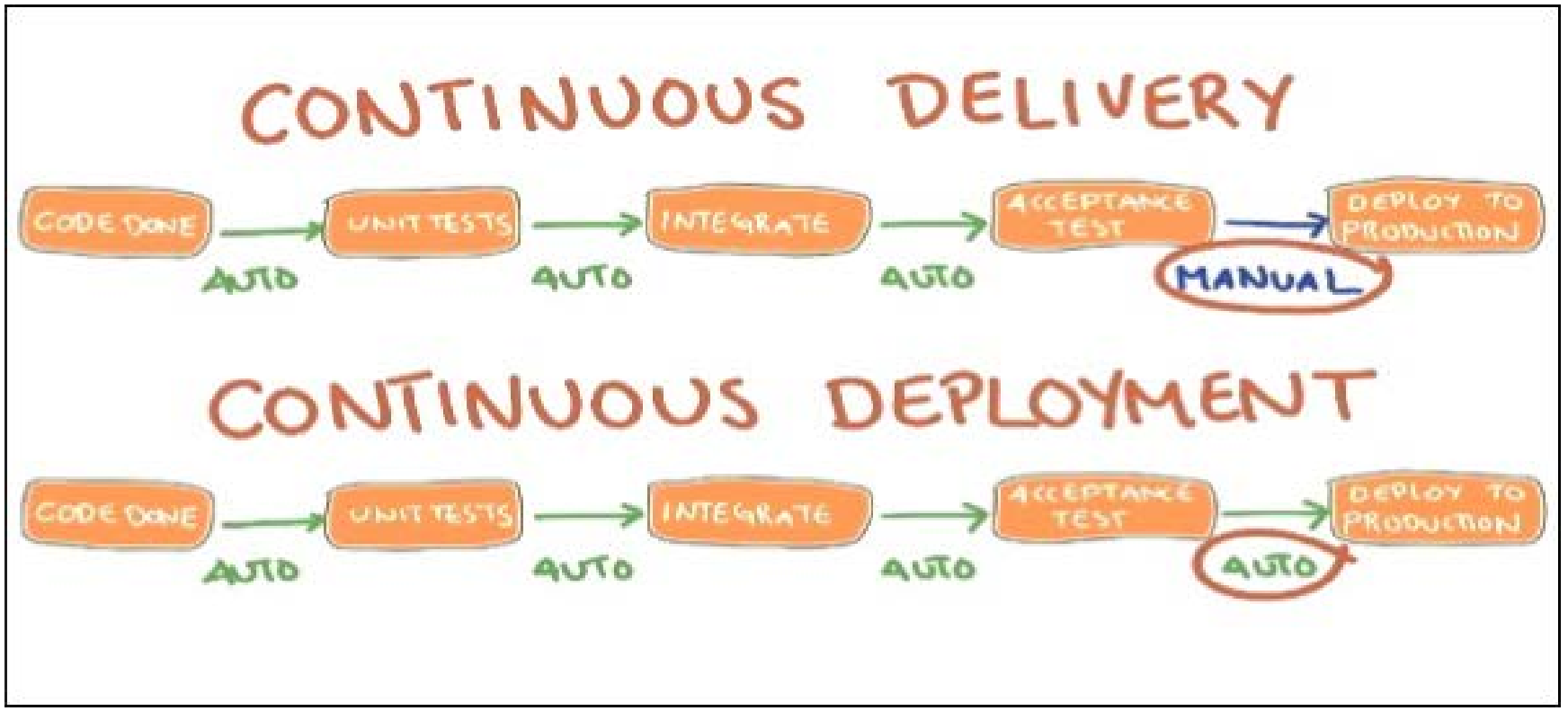
- Since testing begins early and is executed continuously, application risks are exposed soon after they are introduced.
- Development teams can then prevent those problems from progressing to the next stage of the SDLC.
- This reduces the time and effort that need to be spent finding and fixing defects.
- As a result, it is possible to increase the speed and frequency at which quality software (software that meets expectations for an acceptable level of risk) is delivered.

Continuous Deployment

- With Continuous Delivery our software is **always release-ready**, yet the timing of when to push it into production is a business decision.

- With Continuous Deployment, any updated working version of the application is **automatically pushed** to production.

Continuous deployment is the next step of continuous delivery: Every change that passes the automated tests is deployed to production automatically.



- Continuous deployment can be thought of as an extension of continuous integration, aiming at minimizing **lead time**, the time elapsed between development writing one new line of code and this new code being used by live users, in production.

- To achieve continuous deployment, the team relies on infrastructure that automates and instruments the various steps leading up to deployment, so that after each integration successfully meeting these release criteria, the live application is updated with new code.

- **Benefits**
- The main benefits claimed for continuous deployment arise as a result of reducing lead time, with two main effects:

- Earlier return on investment for each feature after it is developed, which reduces the need for large capital investments.

- earlier feedback from users on each new feature as it is released to production.