Module 5: Cloud development and the new IT

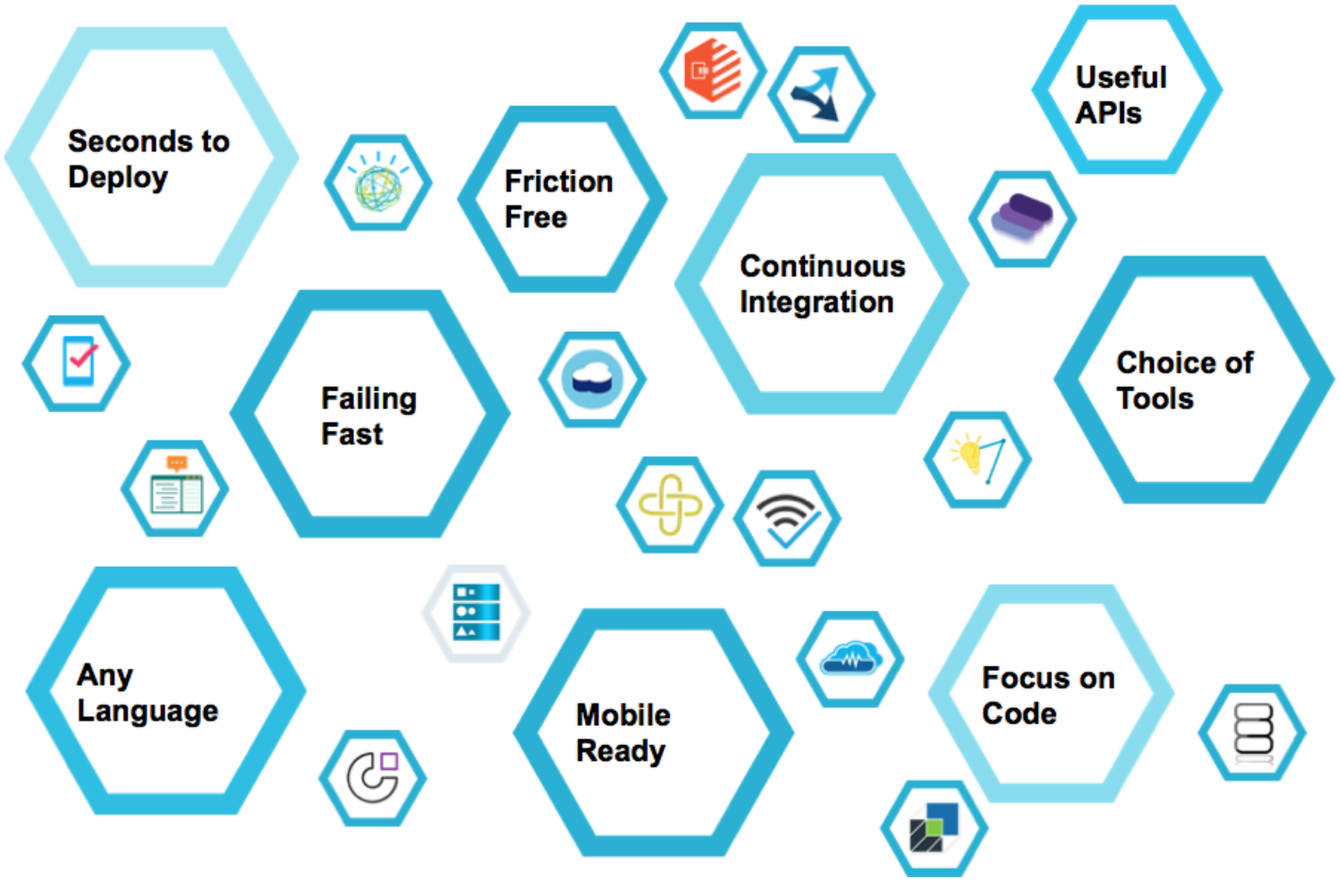
Maximizing the value of Bluemix

Challenges and expectations

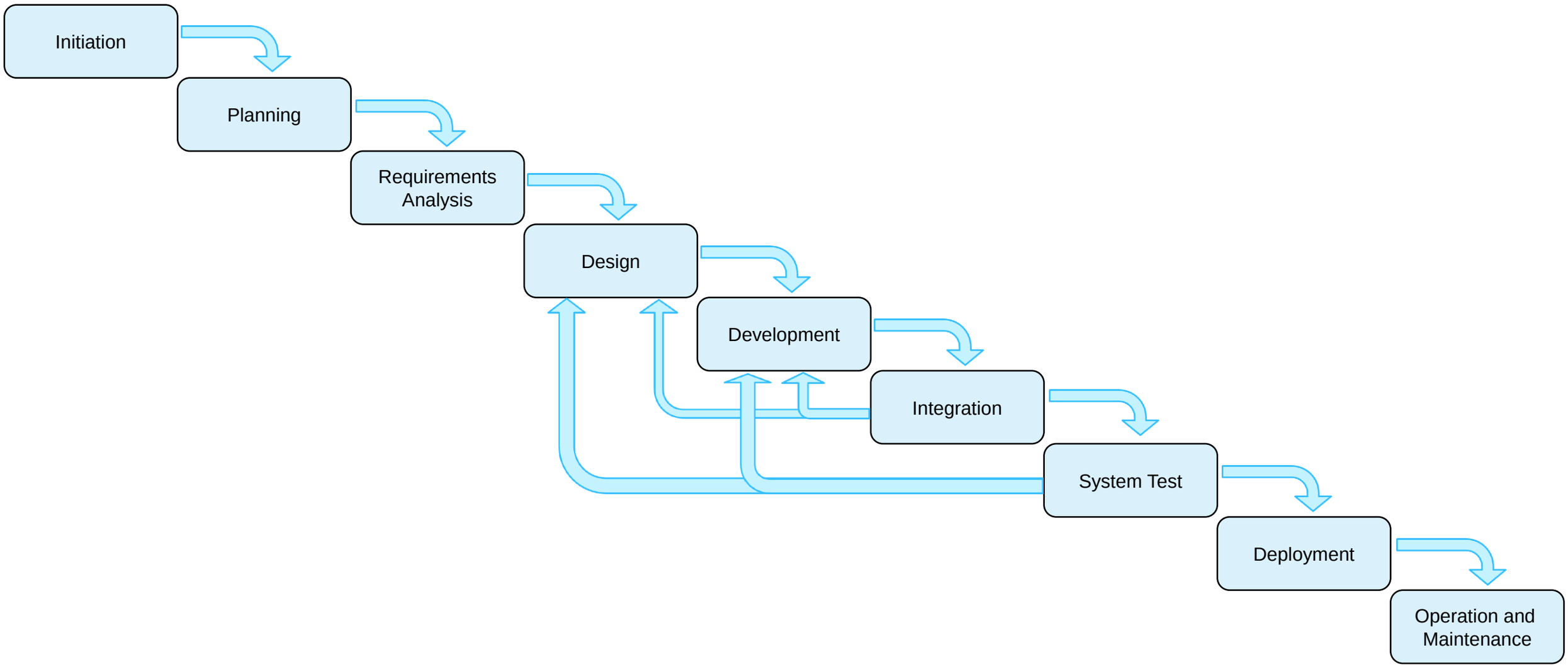
Client business challenges:

- Time to market for new applications is to long
- Speed and innovation are needed to capture new business opportunities
- Remove blockage from IT deployment
- Competitive threat from new ”on the web born” companies
- Clients want to enter the API economy. Need environment to share or sell software assets they build/own
- Reduce operational cost and limit capital investments and remove the need to manage and procure assets and services

Developers’ expectations:

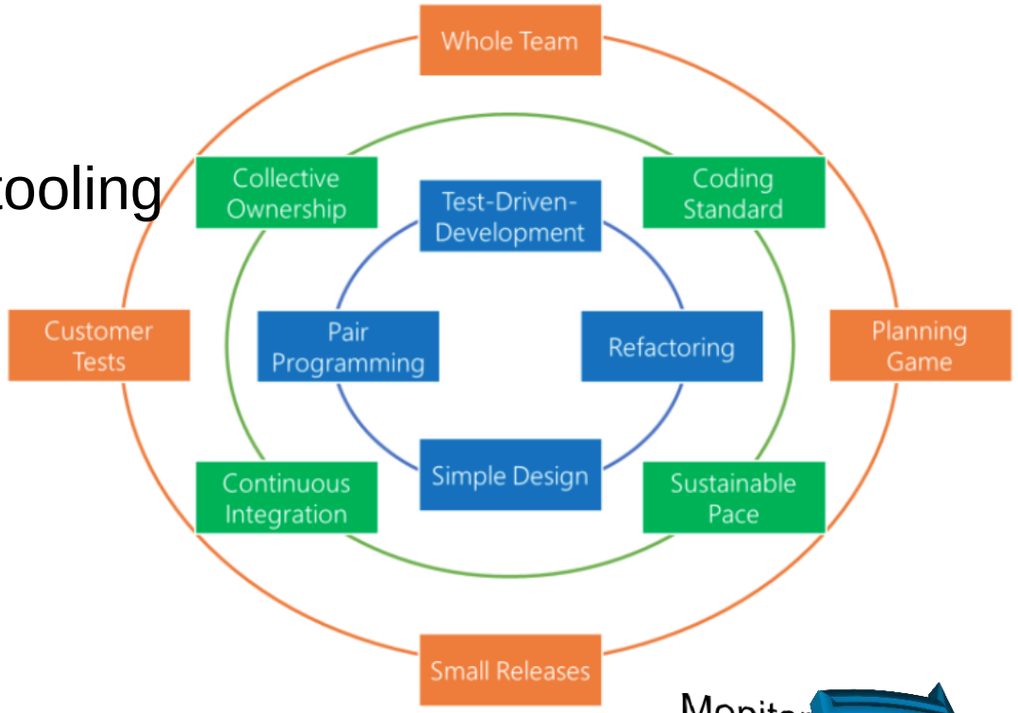


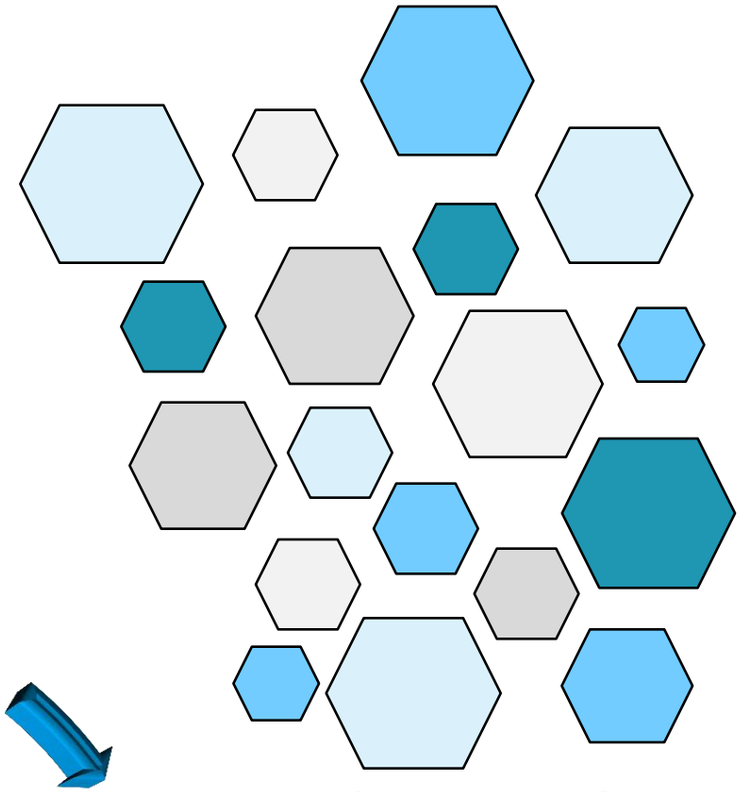
Bluemix can fit into a software development process, but will it deliver the promised value?



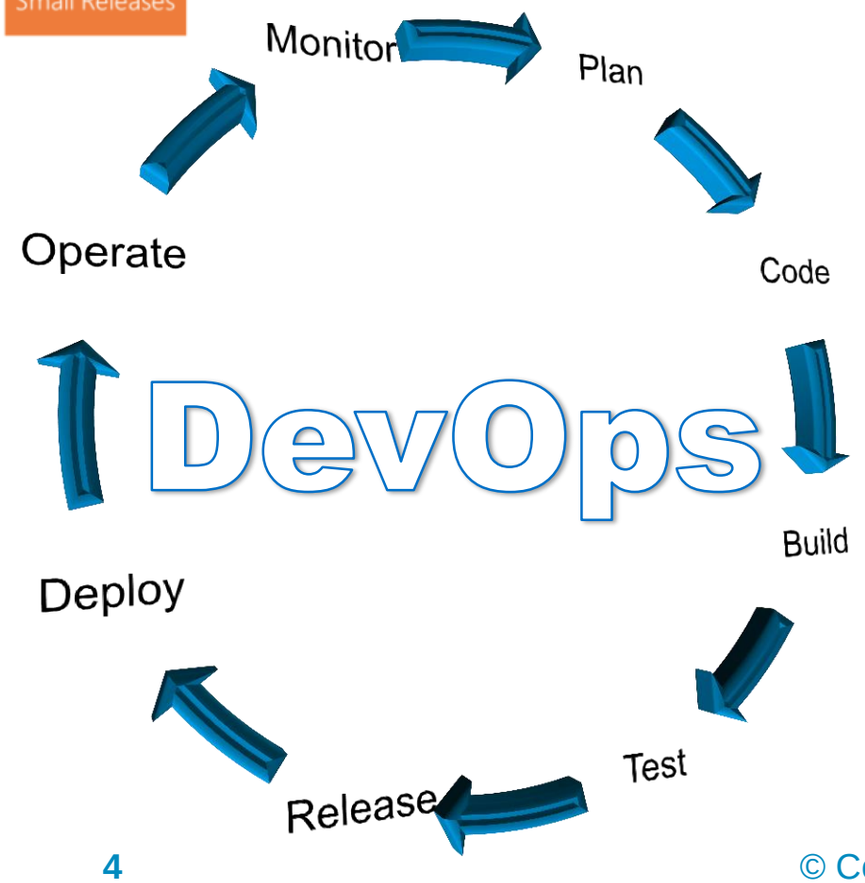
What does it take to deliver the promised value?

- User-centered design
- Agile methodology
- DevOps processes and tooling
- Architect for cloud

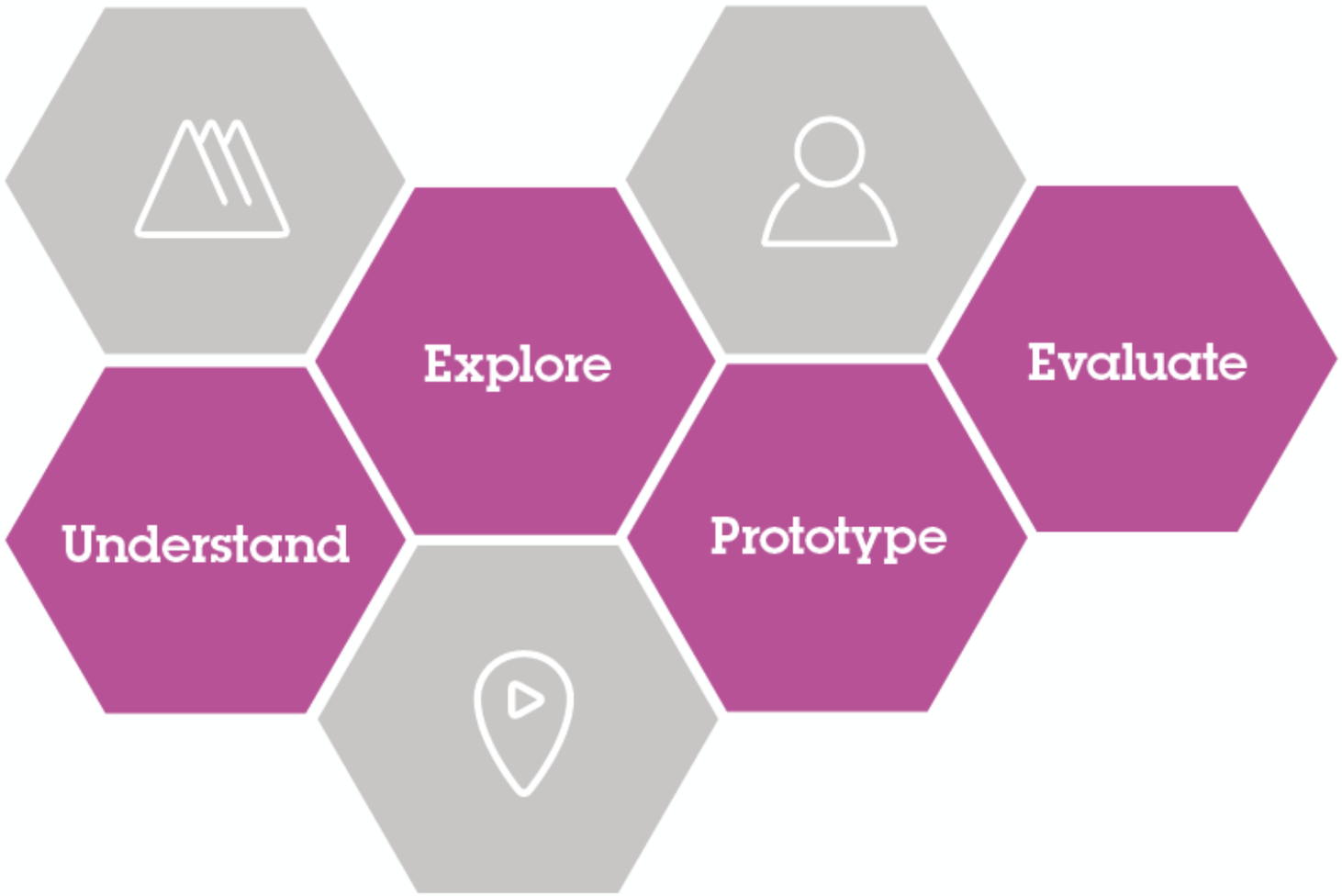




Microservices Architecture



User-centered design



Design thinking methods help you envision your user experience.

Hills: who, what, WOW

Sponsored users: design for real users, not for imagined needs

Playbacks: collaborate to tell a great story

Agile methodology

From Wikipedia:

Agile principles

The Agile Manifesto is based on 12 principles:

•Customer satisfaction by rapid delivery of useful software
•Welcome changing requirements, even late in development
•Working software is delivered frequently
•Close, daily cooperation between business people and developers
•Projects are built around motivated individuals, who should be trusted
•Face-to-face conversation is the best form of communication (co-location)
•Working software is the principal measure of progress
•Sustainable development, able to maintain a constant pace
•Continuous attention to technical excellence and good design
•Simplicity—the art of maximizing the amount of work not done—is essential
•Self-organizing teams
•Regular adaptation to changing circumstance

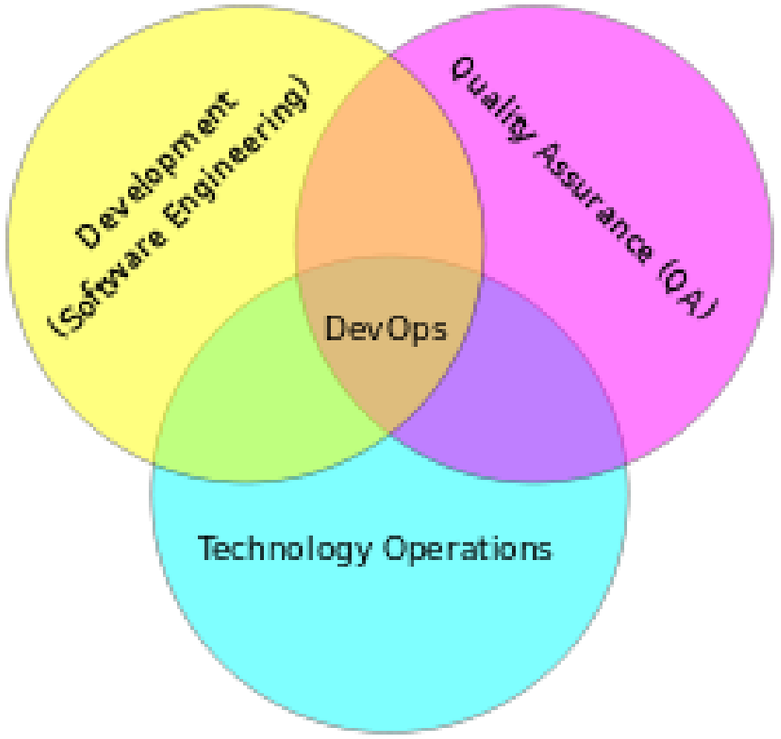
Agile practices

Agile development is supported by several concrete practices, covering areas such as requirements, design, modeling, coding, testing, project management, process, and quality. Some notable agile practices include:

- Acceptance test-driven development (ATDD)
- Agile modeling
- Backlogs (product and sprint)
- Behavior-driven development (BDD)
- Cross-functional team
- Continuous integration (CI)
- Domain-driven design (DDD)
- Information radiators (scrum board, task board, visual management board, burndown chart)
- Iterative and incremental development (IID)
- Pair programming
- Planning poker
- Refactoring
- Scrum events (sprint planning, daily scrum, sprint review, and retrospective)
- Test-driven development (TDD)
- Agile testing
- Timeboxing
- Use case
- User story
- Story-driven modeling
- Retrospective
- Velocity tracking

DevOps

From Wikipedia:



Goals

The specific goals of a DevOps approach span the entire delivery pipeline. They include improved deployment frequency, which can lead to faster time to market, lower failure rate of new releases, shortened lead time between fixes, and faster mean time to recovery in the event of a new release crashing or otherwise disabling the current system.

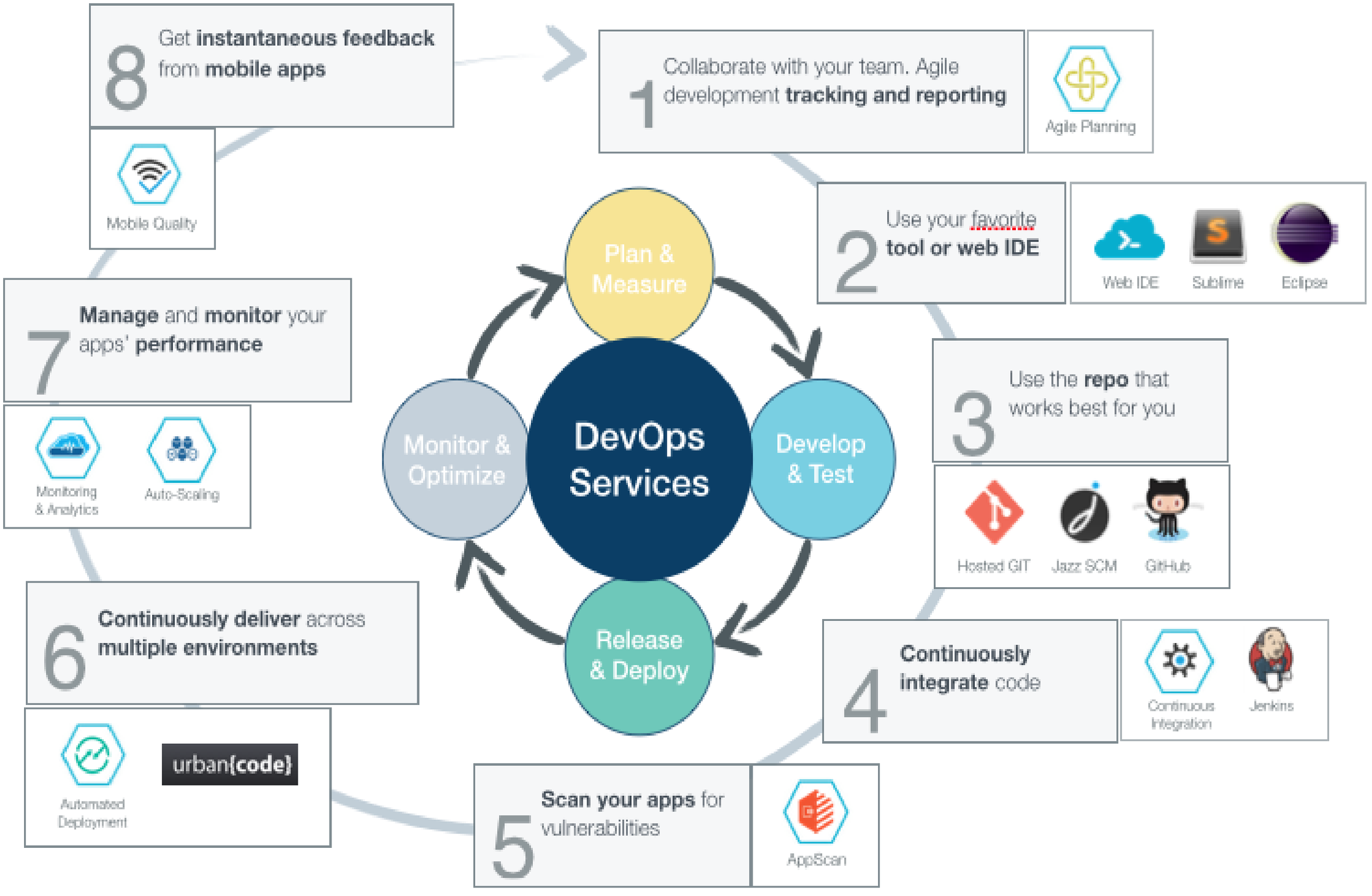
Simple processes become increasingly programmable and dynamic, using a DevOps approach, which aims to maximize the predictability, efficiency, security, and maintainability of operational processes. Very often, automation supports this objective.

Adoption

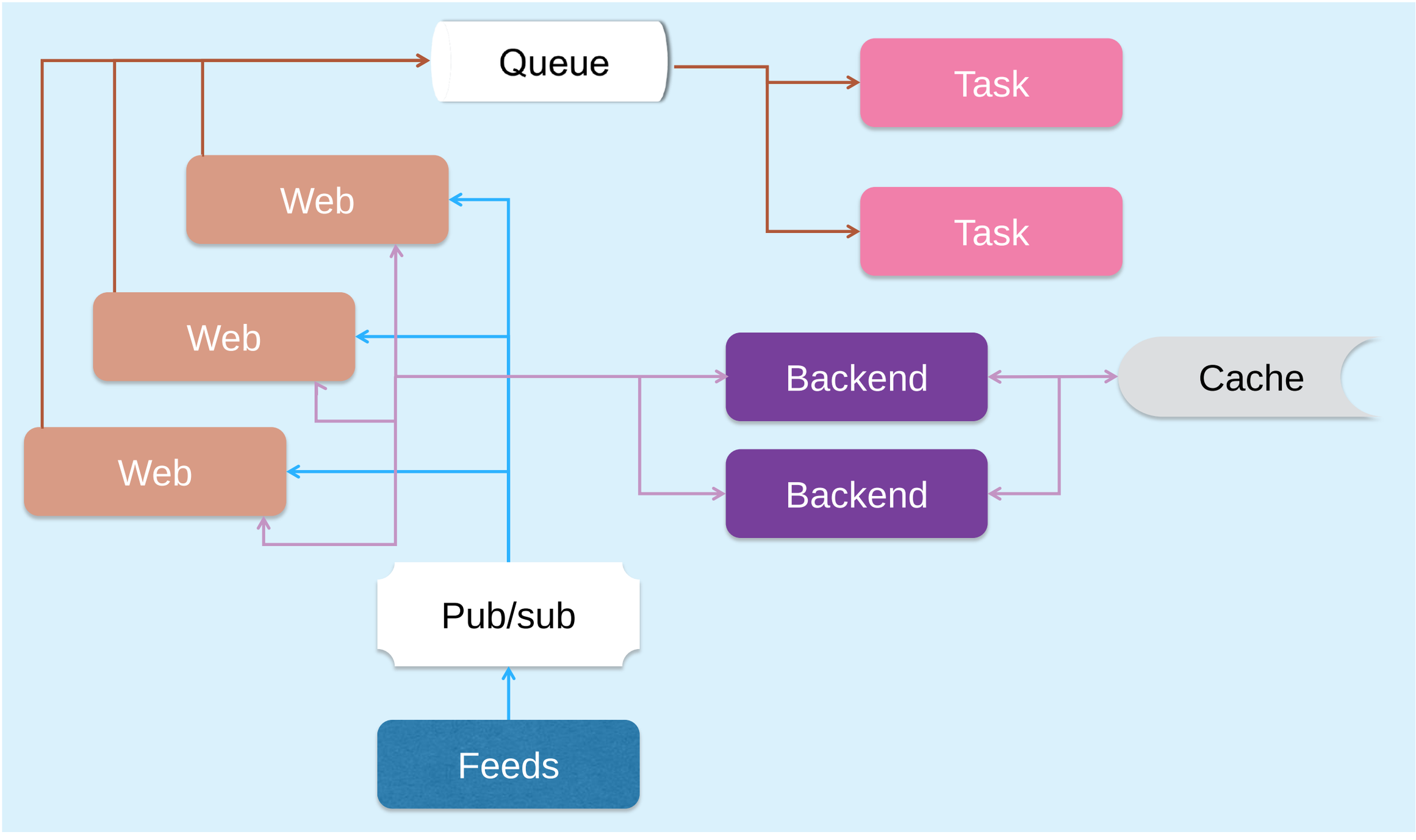
The adoption of DevOps is being driven by factors such as:

- Use of agile and other development processes and methodologies
- Demand for an increased rate of production releases from application and business unit stakeholders
- Wide availability of virtualized and cloud infrastructure from internal and external providers
- Increased usage of data center automation and configuration management tools

Bluemix unified DevOps experience (roadmap)



Application architecture



Twelve-Factor app
(http://12factor.net)

In the modern era, software is commonly delivered as a service: called *web apps*, or *software-as-a-service*.
The twelve-factor app is a methodology for building software-as-a-service apps that:

- Use declarative formats for setup automation to minimize time and cost for new developers joining the project
- Have a clean contract with the underlying operating system, which offers maximum portability between execution environments
- Are suitable for deployment on modern cloud platforms, obviating the need for servers and systems administration
- Minimize divergence between development and production, enabling continuous deployment for maximum agility
- Can scale up without significant changes to tooling, architecture, or development practices

The twelve-factor methodology can be applied to apps written in any programming language, and which use any combination of backing services (database, queue, memory cache, etc).

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

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

Best tools for the job

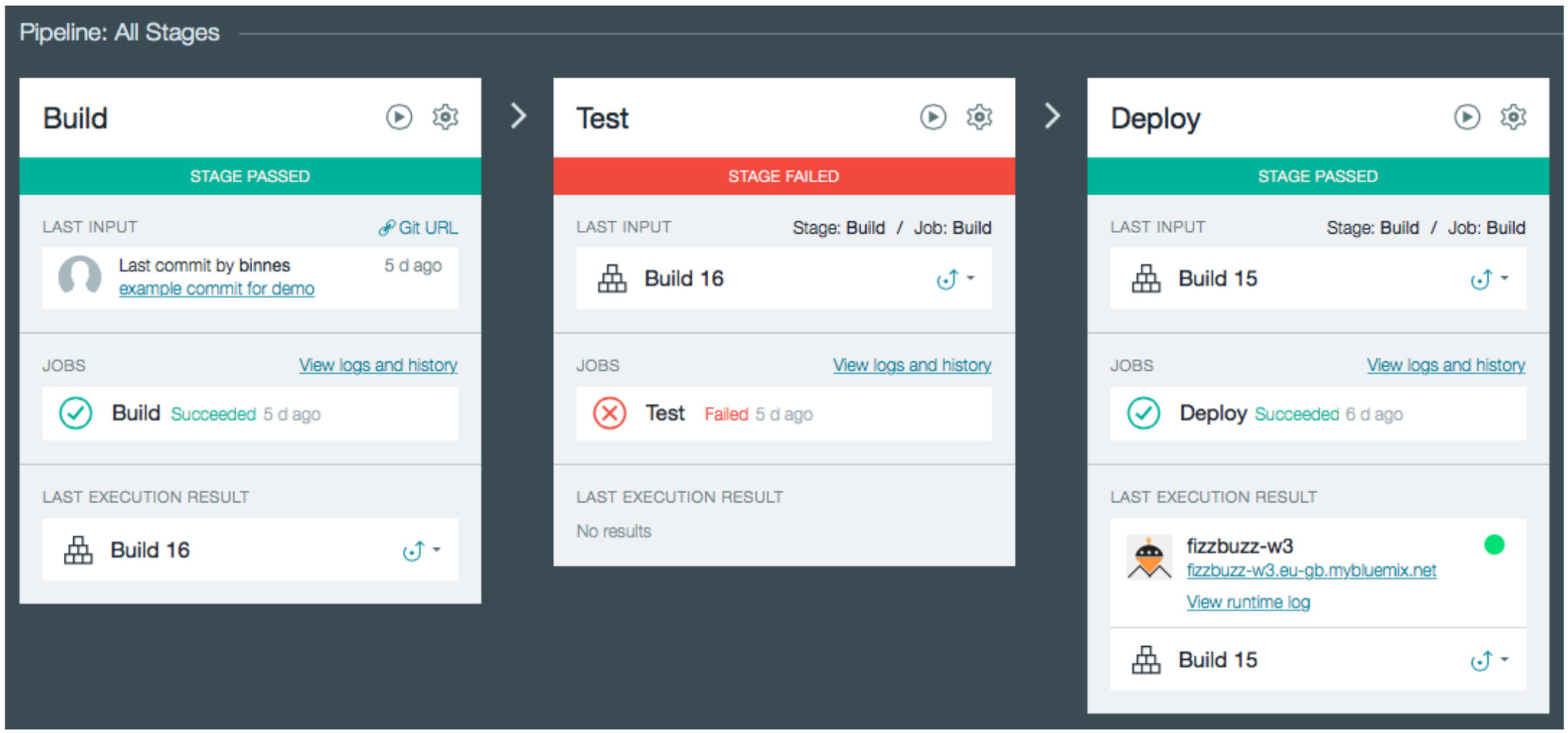
- Importance of having a good developer setup
- Lint tooling (static analysis is useful for code quality)
- Syntax highlighting, autocomplete, etc.
- Quality Assurance needs to be part of the development process
  - Continuous testing during development
  - Test tools such as Mocha, Chai, Sinon for JavaScript

Test-driven development

- No code is written unless it is to pass a test.
- When one function calls another function, ensure that unit tests are not duplicating tests done by the tests for the called function.
  - Test frameworks provide *spy* functionality to verify functions are called correctly rather than testing what the called function returns.
- Wherever possible, ensure tests cover problematic areas of the implementation language.
  - For example, with a JavaScript string vs number comparison, “5” > “11” but 5 < 11 type coercion sometimes delivers the wrong comparison

DevOps pipeline in Bluemix

IBM Bluemix DevOps Services can be configured to automatically run tests and if the tests pass, the code is deployed to Bluemix.



Summary

- A modern cloud platform alone does not deliver the agility and speed businesses require today.
- A modern approach to application creation is needed, which delivers:
  - Innovation
  - Agility
  - Quality
  - Best user experience
- This approach requires an appropriate design, methodology, set of processes, tooling, and architecture to enable developers to deliver.
- Many companies are adopting a bimodal approach to manage existing applications and enable innovative new applications to be created.