Desenvolvimento de Aplicações com Arquitetura Baseada em Microservices

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: @vinicius3w :: assertlab.com

[IF1007] - Tópicos Avançados em SI 4
https://github.com/vinicius3w/if1007-Microservices

Licença do material

Este Trabalho foi licenciado com uma Licença

Creative Commons - Atribuição-NãoComercial-CompartilhaIgual 3.0 Não Adaptada

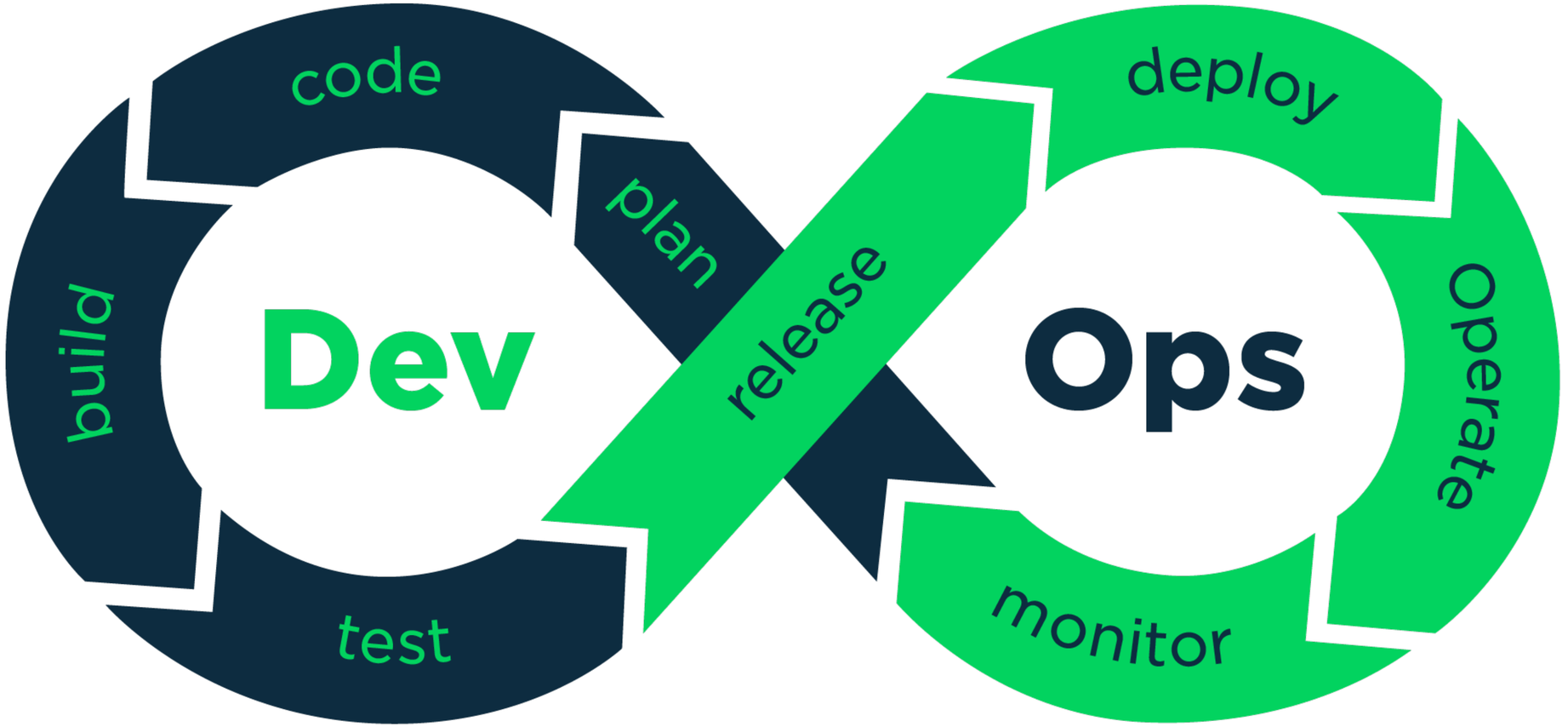
Mais informações visite

http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt

AssertLab
Advanced Software and Systems Engineering Research Technologies

Resources

- There is no textbook required. However, the following are some books that may be recommended:
  - Introduction to Kubernetes - LinuxFoundationX - LFS158x
  - Kubernetes Ingress: NodePort, Load Balancers, and Ingress Controllers
  - Deploying Java Applications with Docker and Kubernetes
  - Deploying Java Applications with Kubernetes and an API Gateway
  - Kubernetes Documentation
  -



The Microservices Development Life Cycle

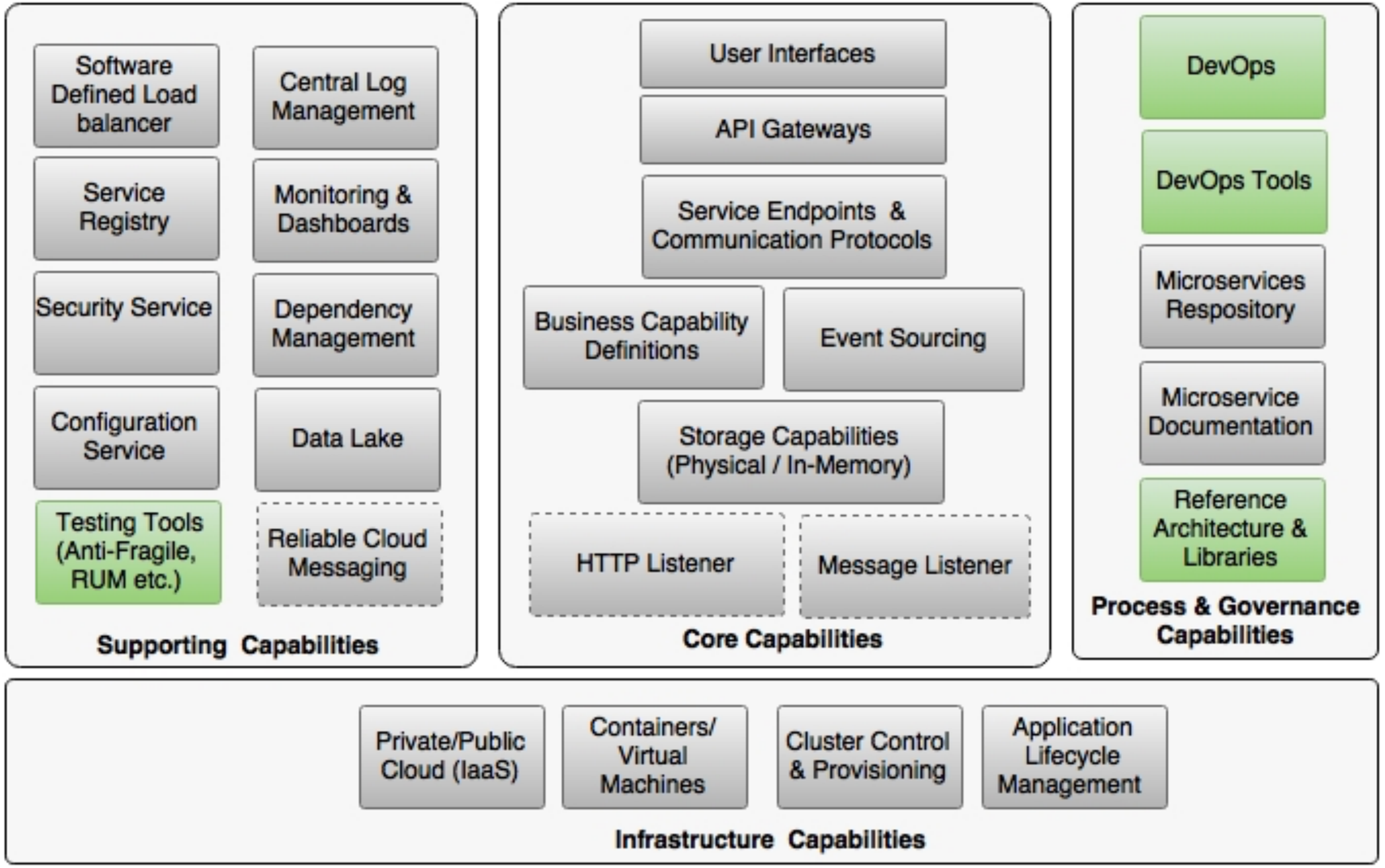
Overview

- Similar to the software development life cycle (SDLC), it is important to understand the aspects of the microservice development life cycle processes for a successful implementation of the microservices architecture
- What are the best practices in structuring development teams, development methodologies, automated testing, and continuous delivery of microservices in line with DevOps practices?
- What is the importance of the reference architecture in a decentralized governance approach to microservices?

Topics

- Reviewing DevOps in the context of microservices development
- Defining the microservices life cycle and related processes
- Best practices around the development, testing, and deployment of Internet-scale microservices

Reviewing the microservice capability model



"DevOps represents a change in IT culture, focusing on rapid IT service delivery through the adoption of agile, lean practices in the context of a system-oriented approach. DevOps emphasizes people (and culture), and seeks to improve collaboration between operations and development teams. DevOps implementations utilize technology — especially automation tools that can leverage an increasingly programmable and dynamic infrastructure from a life cycle perspective."

–Gartner

The new mantra of lean IT – DevOps

- DevOps and microservices evolved independently
- In this section, we will review the evolution of DevOps and then take a look at how DevOps supports microservices adoption

The new mantra of lean IT – DevOps

- IT organizations have to master two key areas: speed of delivery and value-driven delivery
- Many IT organizations failed to master this change
  - To overcome this situation, many business departments started their own shadow IT or stealth IT under their control
  - Some smart IT organizations then adopted a lean IT model to respond to these situations

The new mantra of lean IT – DevOps

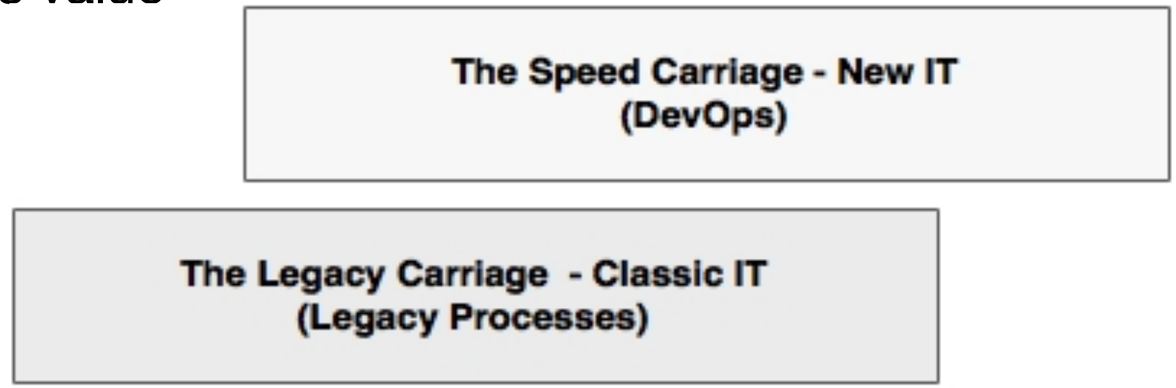
- However, many organizations still struggle with this transformation due to the large baggage of legacy systems and processes
- Gartner coined the concept of a **pace-layered application** strategy
  - high speed is required only for certain types of applications or certain business areas
  - Gartner termed this a **system of innovation**
    - A system of innovation requires rapid innovations compared to a **system of records**
- As a system of innovations needs **rapid innovation**, a **lean IT delivery model** is essential for such applications
  - Practitioners evangelized the lean IT model as **DevOps**

The new mantra of lean IT – DevOps

- There are two key strategies used by organizations to adopt DevOps
  - Some organizations positioned DevOps as a process to fill the gaps in their existing processes
    - An incremental strategy for their DevOps journey
    - Starts with Agile development, then incrementally adopts continuous integration, automated testing, and release to production and then all DevOps practices
    - The challenge is the time to realize the full benefits as well as the mixed culture of people due to legacy processes

The new mantra of lean IT – DevOps

- There are two key strategies used by organizations to adopt DevOps
  - Many organizations, therefore, take a disruptive approach to adopt DevOps
    - This will be achieved by partitioning IT into two layers or even as two different IT units
      - The high-speed layer of IT uses DevOps-style practices to dramatically change the culture of the organization with no connection to the legacy processes and practices
      - A selective application cluster will be identified and moved to the new IT based on the business value



Reducing wastage

- DevOps processes and practices essentially speed up deliveries which improves quality
- The speed of delivery is achieved by cutting IT wastage
- This is achieved by avoiding work that adds no value to the business nor to desired business outcomes
- The wastage is reduced by primarily adopting Agile processes, tools, and techniques

Automating every possible step

- By automating the manually executed tasks, one can dramatically improve the speed of delivery as well as the quality of deliverables
- This also reduces a number of manual gate checks, bureaucratic decision making, and so on
- Automated monitoring mechanisms and feedback go back to the development factory, which gets it fixed and quickly moved to production.

Value-driven delivery

- DevOps reduces the gap between IT and business through value-driven delivery
- Value-driven delivery closely aligns IT to business by understanding **true business values** and helps the business by **quickly delivering these values**, which can give a competitive advantage
  - This is similar to the shadow IT concept, in which IT is collocated with the business and delivers business needs quickly, rather than waiting for heavy project investment-delivery cycles

Bridging development and operations

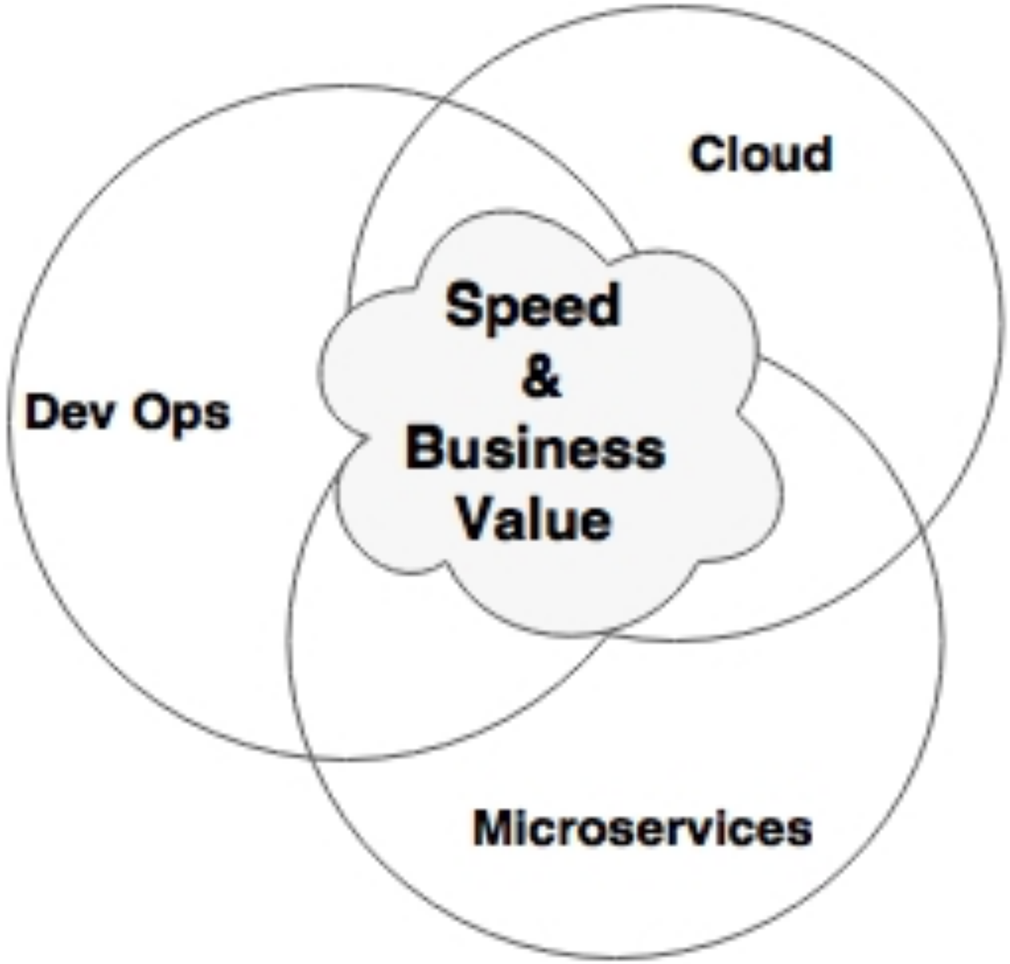
- DevOps reduces the gap between the development and operations teams so that it can potentially reduce wastage and improve quality
  - Multidisciplinary teams work together to address problems at hand rather than throwing mud across the wall
  - Operations teams can make decisions based **exactly on service behaviors** rather than **enforcing standard organizational policies and rules** when designing infrastructure components
  - This would eventually help the IT organization to improve the quality of the product as well as the time to resolve incidents and problem management

In the DevOps world…

- Speed of delivery is achieved through the automation of high-velocity changes
- Quality is achieved through automation and people
- Business values are achieved through efficiency, speed of delivery, quality, and the ability to innovate
- Cost reduction is achieved through automation, productivity, and reducing wastage

Meeting the trio – microservices, DevOps, and cloud

- Targets a set of common objectives: speed of delivery, business value, and cost benefit
- All three can stay and evolve independently, but they complement each other to achieve the desired common goals
- Organizations embarking on any of these naturally tend to consider the other two as they are closely linked together



Cloud as the self-service infrastructure for Microservices

- The main driver for cloud is to improve agility and reduce cost
  - By reducing the time to provision the infrastructure, the speed of delivery can be increased
  - By optimally utilizing the infrastructure, one can bring down the cost
  - Therefore, cloud directly helps achieve both speed of delivery and cost
- Without having a cloud infrastructure with cluster management software, it would be hard to control the infrastructure cost when deploying microservices
  - infrastructure as code or software-defined infrastructure

DevOps as the practice and process for microservices

- Microservice is an architecture style that enables quick delivery
  - However, microservices cannot provide the desired benefits by themselves
  - Microservices need a set of supporting delivery practices and processes to effectively achieve their goal
- DevOps is the ideal candidate for the underpinning process and practices for microservice delivery
  - DevOps processes and practices gel well with the microservices architecture's philosophies

Practice points for microservices development

- For a successful microservice delivery, a number of **development-to-delivery practices** need to be considered, including the DevOps philosophy
- In the previous sections, you learned the different architecture capabilities of microservices
- In this section, we will explore the nonarchitectural aspects of microservice developments

Understanding business motivation and value

- Microservices should not be used for the sake of implementing a niche architecture style
- It is extremely important to understand the business value and business KPIs before selecting microservices as an architectural solution for a given problem
- A good understanding of business motivation and business value will help engineers focus on achieving these goals in a cost-effective way
- Business motivation and value should justify the selection of microservices

Changing the mindset from project to product development

- Microservices are more aligned to product development
  - Business capabilities that are delivered using microservices should be treated as products
  - This is in line with the DevOps philosophy as well

Changing the mindset from project to product development

- The product team will always have a sense of ownership and take responsibility for what they produce
- As a result, product teams always try to improve the quality of the product
- The product team is responsible not only for delivering the software but also for production support and maintenance of the product
- Product teams are generally linked directly to a business department for which they are developing the product
- Product thinking is closely aligned with actual business goals

Changing the mindset from project to product development

- One common pitfall in product development is that IT people represent the business in the product team
- These IT representatives may not fully understand the business vision
- Also, they may not be empowered to take decisions on behalf of the business
- Such cases can result in a misalignment with the business and lead to failure quite rapidly

Choosing a development philosophy

- Different organizations take different approaches to developing microservices, be it a migration or a new development
- It is important to choose an approach that suits the organization
- There is a wide verity of approaches available

Design thinking

- DT is an approach primarily used for innovation-centric development
  - Explores the system from an end user point of view: what the customers see and how they experience the solution
  - A story is then built based on observations, patterns, intuition, and interviews
- DT then quickly devises solutions through solution-focused thinking by employing a number of theories, logical reasoning, and assumptions around the problem
  - The concepts are expanded through brainstorming before arriving at a converged solution

The start-up model

- Many start-ups kick off with a small, focused team—a highly cohesive unit
  - The unit is not worried about how they achieve things; rather, the focus is on what they want to achieve
  - Once they have a product in place, the team thinks about the right way to build and scale it
- This approach addresses quick delivery through production-first thinking
  - The advantage with this approach is that teams are not disturbed by organizational governance and political challenges
  - The team is empowered to think out of the box, be innovative, and deliver things
  - They also follow a fail fast approach and course correct sooner than later

The Agile practice

- Software is delivered in an incremental, iterative way using the principles put forth in the Agile manifesto
- This type of development uses an Agile method such as Scrum or XP
- The Agile manifesto defines four key points that Agile software development teams should focus on:
  - Individuals and interaction over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Using the concept of Minimum Viable Product

- Irrespective of the development philosophy explained earlier, it is essential to identify a **Minimum Viable Product (MVP)** when developing microservice systems for speed and agility
-

“A Minimum Viable Product is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort.”

–Eric Ries

Using the concept of Minimum Viable Product

- The objective of the MVP approach is to **quickly** build a piece of software that **showcases** the **most important aspects** of the software
- The MVP approach realizes the **core concept** of an idea and perhaps chooses those features that **add maximum value** to the business
  - It helps get early feedback and then course corrects as necessary before building a heavy product

Overcoming the legacy hotspot

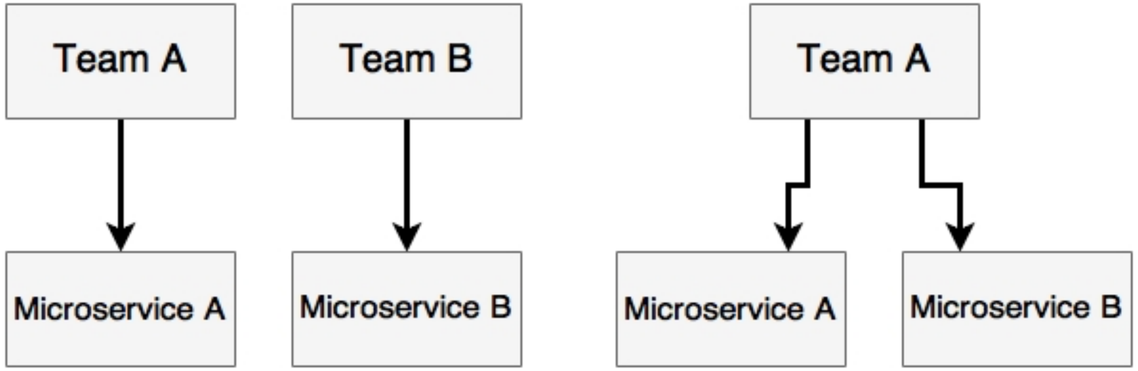
- It is important to understand the environmental and political challenges in an organization before embarking on microservices development
- It is common in microservices to have dependencies on other legacy applications, directly or indirectly
- A common issue with direct legacy integration is the slow development cycle of the legacy application
- This is especially common when migrating legacy monolithic applications to microservices
  - In many cases, legacy systems continue to undergo development in a non-Agile way with larger release cycles

Addressing challenges around databases

- Automation is key in microservices development
  - Automating databases is one of the key challenges in many microservice developments
- Many automation tools focus on the application logic
  - As a result, many development teams completely ignore database automation
  - Ignoring database automation can severely impact the overall benefits and can derail microservices development
- The database has to be treated in the same way as applications with appropriate source controls and change management
  - When selecting a database, it is also important to consider automation as one of the key aspects
- **Database automation** is much easier in the case of NoSQL databases but is hard to manage with traditional RDBMs
  - **Database Lifecycle Management (DLM)** as a concept is popular in the DevOps world, particularly to handle database automation
  - Tools such as DBmaestro, Redgate DLM, Datical DB, and Delphix support database automation

Establishing self-organizing teams

- One of the most important activities in microservices development is to establish the right teams for development
- As recommended in many DevOps processes, a small, focused team always delivers the best results

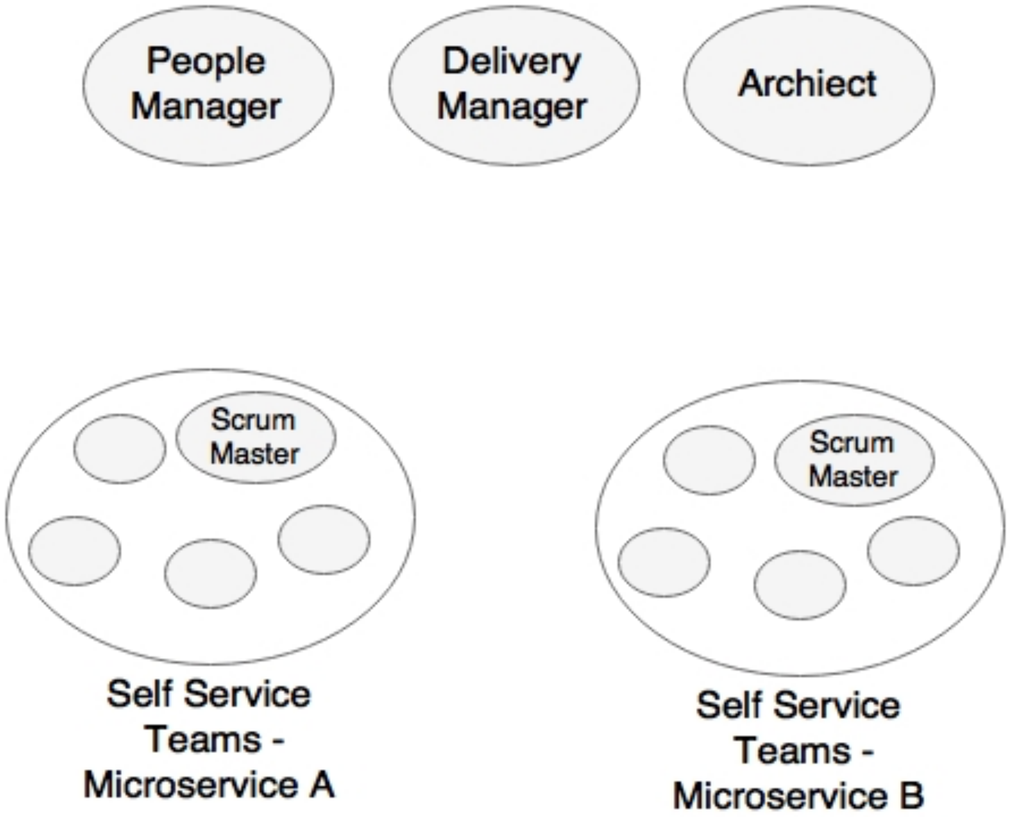


Establishing self-organizing teams

- Teams are expected to take full ownership in ideating for, analyzing, developing, and supporting services
  - Werner Vogels from Amazon.com calls this you build it and you run it
  - As per Werner's theory, developers pay more attention to develop quality code to avoid unexpected support calls
- Teams should have multidisciplinary skills to satisfy all the capabilities required to deliver a service
- One common solution to this problem is to use the concept of consultants
  - Consultants are SMEs and are engaged to gain expertise on specific problems faced by the team
  - Some organizations also use shared or platform teams to deliver some common capabilities

Establishing self-organizing teams

- Agile software development also encourages having self-organizing teams
- Self-organizing teams act as a cohesive unit and find ways to achieve their goals as a team
- The team automatically align themselves and distribute the responsibilities
- The members in the team are self-managed and empowered to make decisions in their day-to-day work
- The team's communication and transparency are extremely important in such teams
  - This emphasizes the need for collocation and collaboration, with a high bandwidth for communication



Building a self-service cloud

- What microservice developers need is more than just an IaaS cloud platform
  - Neither the developers nor the operations engineers in the team should worry about where the application is deployed and how optimally it is deployed
  - They also should not worry about how the capacity is managed
- This level of sophistication requires a cloud platform with self-service capabilities (as we discussed earlier)
  - Mesos and Marathon and Kubernetes cluster management solutions
  - Containerized deployment is also important in managing and end to-end-automation
  - Building this self-service cloud ecosystem is a prerequisite for microservice development

Building a microservices ecosystem

- Microservices require a number of capabilities
  - All these capabilities should be in place before implementing microservices at scale
- These capabilities include service registration, discovery, API gateways, and an externalized configuration service
  - All are provided by the Spring Cloud project
  - Capabilities such as centralized logging, monitoring, and so on are also required as a prerequisite for microservices development

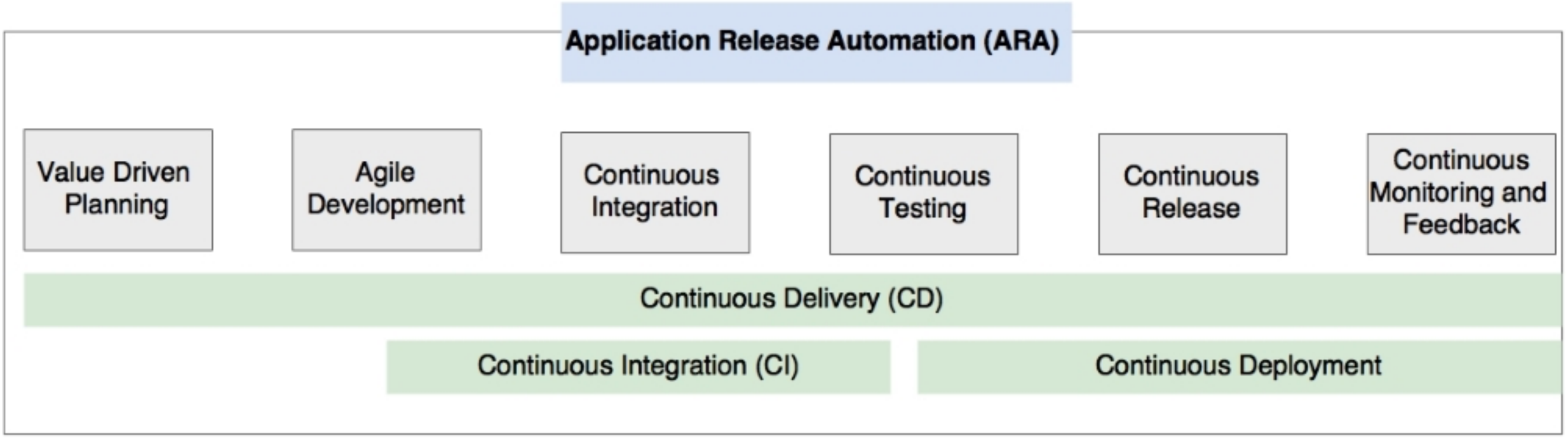
Defining a DevOps-style microservice life cycle process

- Organizations already practicing DevOps do not need another practice for microservices development
- Rather than reinventing a process for microservices, we will explore DevOps processes and practices from the microservice perspective

Common terminologies used in the DevOps world

- **Continuous integration (CI)**: This automates the application build and quality checks continuously in a designated environment, either in a time-triggered manner or on developer commits
- **Continuous delivery (CD)**: This automates the end-to-end software delivery practice from idea to production
  - In a non-DevOps model, this used to be known as **Application Lifecycle Management (ALM)**
- **Continuous deployment**: This is an approach to automating the deployment of application binaries to one or more environments by managing binary movement and associated configuration parameters
- **Application Release Automation (ARA)**: ARA tools help monitor and manage end-to-end delivery pipelines. ARA tools use CI and CD tools and manage the additional steps of release management approvals

DevOps process for microservices development



Value-driven planning

- Value-driven planning is a term used in Agile development practices
  - Identify which microservices to develop
- The most important aspect is to identify those requirements that have the highest value to business and those that have the lowest risks
- The MVP philosophy is used when developing microservices from the ground up
- The selected microservices are expected to precisely deliver the expected value to the business
- Business KPIs to measure this value have to be identified as part of value-driven planning

Agile development

- Once the microservices are identified, development must be carried out in an Agile approach following the Agile manifesto principles, for instance
- The scrum methodology is used by most of the organizations for microservices development

Continuous integration

- The continuous integration steps should be in place to automatically build the source code produced by various team members and generate binaries
- Continuous integration also executes various QAs as part of the build pipeline, such as code coverage, security checks, design guidelines, and unit test cases

Continuous testing

- Once continuous integration generates the binaries, they are moved to the testing phase
- A fully automated testing cycle is kicked off in this phase
- This could range from the integration test environment to the production environment to test in production

Continuous release

- Continuous release to production takes care of actual deployment, infrastructure provisioning, and rollout
- The binaries are automatically shipped and deployed to production by applying certain rules
- Many organizations stop automation with the staging environment and make use of manual approval steps to move to production

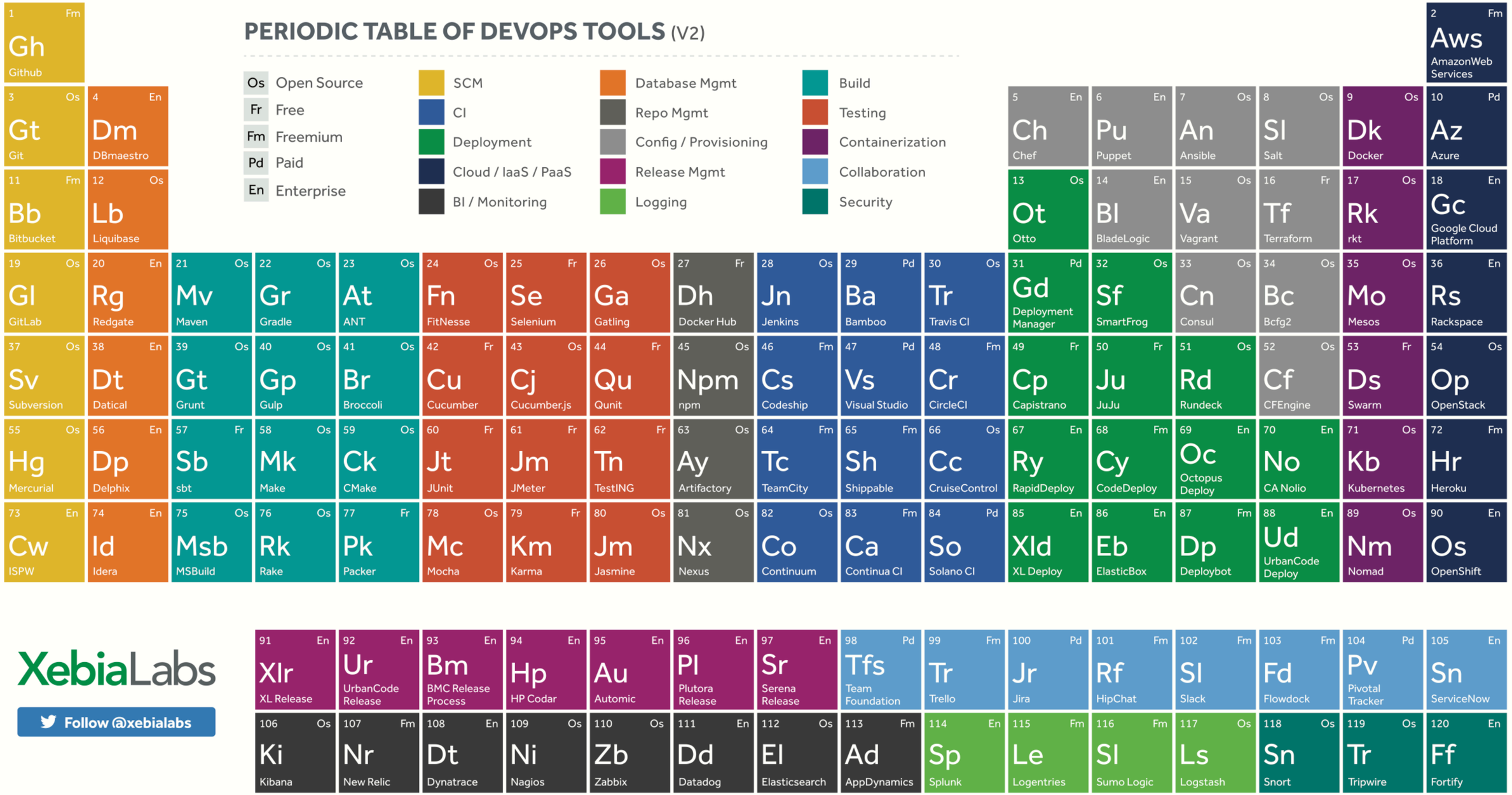
Continuous monitoring and feedback

- The continuous monitoring and feedback phase is the most important phase in Agile microservices development
- Based on the feedback, the services are adjusted and the same cycle is then repeated

Automating the continuous delivery pipeline

- The life cycle stages can be altered by organizations based on their organizational needs but also based on the nature of the application
- There are many tools available to build end-to-end pipelines, both in the open source and commercial space
- Organizations can select the products of their choice to connect pipeline tasks

PERIODIC TABLE OF DEVOPS TOOLS (V2)



XebiaLabs

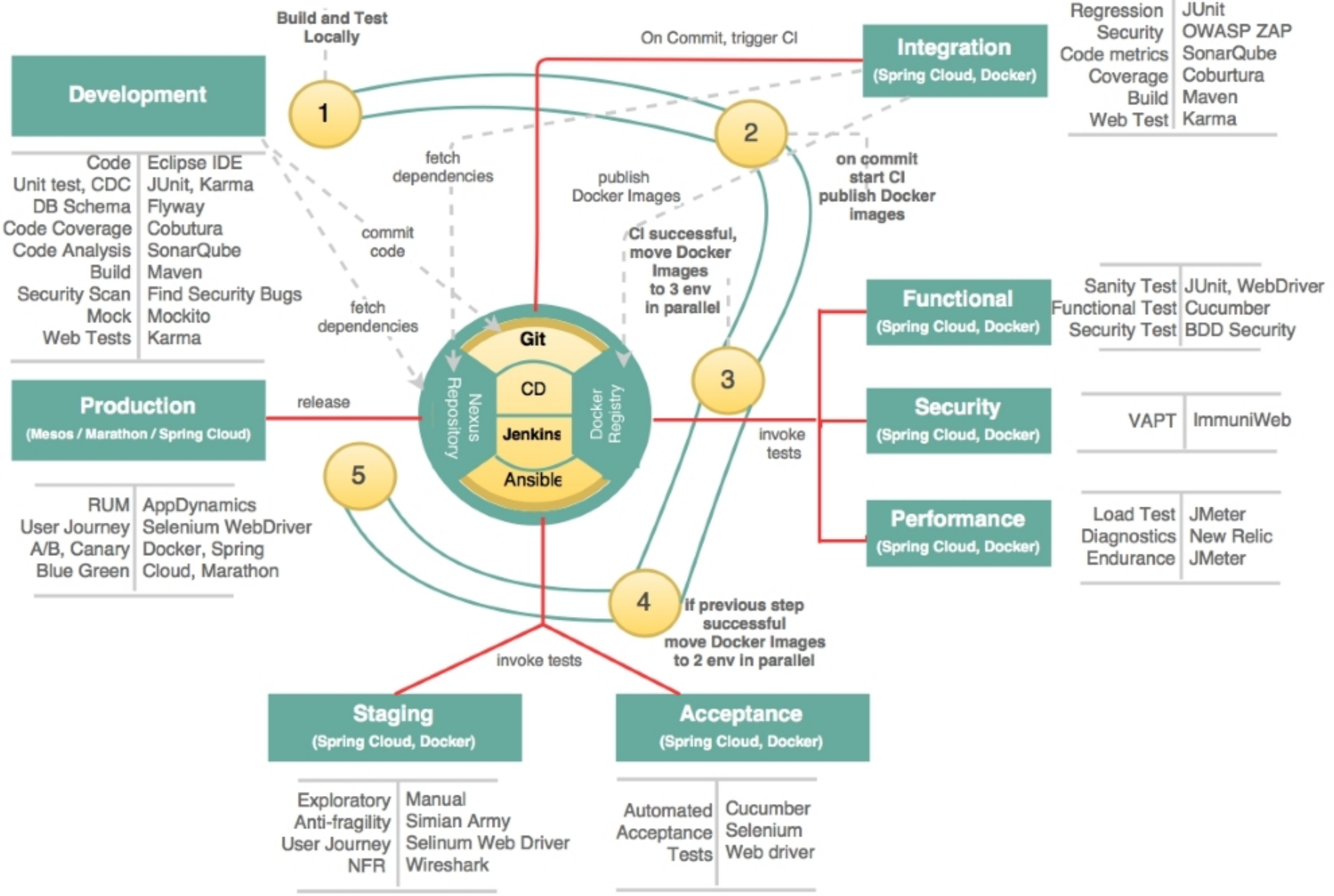
Follow @xebialabs

Automating the continuous delivery pipeline

- The pipelines may initially be expensive to set up as they require many toolsets and environments
- Organizations may not realize an immediate cost benefit in implementing the delivery pipeline
- Also, building a pipeline needs high-power resources
- Large build pipelines may involve hundreds of machines
  - It also takes hours to move changes through the pipeline from one end to the other
  - Hence, it is important to have different pipelines for different microservices

Automating the continuous delivery pipeline

- The key focus in the pipeline is on end-to-end automation, from development to production, and on failing fast if something goes wrong


Build and Test Locally
On Commit, trigger CI
Development
Integration
(Spring Cloud, Docker)
Regression JUnit
Security OWASP ZAP
Code metrics SonarQube
Coverage Coburtura
Build Maven
Web Test Karma
1
2
Code Eclipse IDE
Unit test, CDC JUnit, Karma
DB Schema Flyway
Code Coverage Cobutura
Code Analysis SonarQube
Build Maven
Security Scan Find Security Bugs
Mock Mockito
Web Tests Karma
fetch dependencies
publish Docker Images
on commit start CI publish Docker images
commit code
CI successful, move Docker Images to 3 env in parallel
fetch dependencies
Git
CD
Jenkins
Ansible
Nexus Repository
Docker Registry
3
Functional
(Spring Cloud, Docker)
Sanity Test JUnit, WebDriver
Functional Test Cucumber
Security Test BDD Security
Production
(Mesos / Marathon / Spring Cloud)
release
Security
(Spring Cloud, Docker)
VAPT ImmuniWeb
invoke tests
5
Performance
(Spring Cloud, Docker)
Load Test JMeter
Diagnostics New Relic
Endurance JMeter
RUM AppDynamics
User Journey Selenium WebDriver
A/B, Canary Docker, Spring
Blue Green Cloud, Marathon
4
if previous step successful move Docker Images to 2 env in parallel
invoke tests
Staging
(Spring Cloud, Docker)
Acceptance
(Spring Cloud, Docker)
Exploratory Manual
Anti-fragility Simian Army
User Journey Selinum Web Driver
NFR Wireshark
Automated Acceptance Tests Cucumber Selenium Web driver

Automated configuration management

- Use new methods for configuration management rather than using a traditional statically configured CMDB
  - The manual maintenance of CMDB is no longer an option
- The new styles of CMDB automatically create CI configurations based on an operational topology
  - These should be discovery based to get up-to-date information
  - The new CMDB should be capable of managing bare metals, virtual machines, and containers

Microservices development governance, reference architectures, and libraries

- It is important to have an overall enterprise reference architecture and a standard set of tools for microservices development to ensure that development is done in a consistent manner
- For quick wins and to take advantage of timelines, microservices development teams may deviate from these practices in some cases

Desenvolvimento de Aplicações com Arquitetura Baseada em Microservices

Muito obrigado a todos!

Próximos passos?

AssertLab
Advanced Software and Systems Engineering Research Technologies