IBM Cloud Private Practical Workshop

Application Implementation

DevOps

IBM Cloud Private Use Cases

Characteristics of a **cloud-native** application

Built with the cloud in mind

- Use cloud services
  - E.g. storage, queuing, cache
- Have rapid and repeatable deployments to maximize agility
- Have automated setup to minimize time and cost for new developers
- Have clean contract with underlying OS to ensure maximum portability

Loose ties into corporate IT

- Security often specified using open standards
  - E.g. OpenID, OAuth
- Data might be local to the application itself

QoS attributes are those of the cloud

- It's always expected to be there, but sometimes sites and mobile apps become unavailable, although they must be restored quickly
- Applications must scale elastically without significant changes to tooling, architecture, or development practice
- Application must be resilient to inevitable failures in the infrastructure and application

Cloud-Native Application Goals

Horizontal scaling

- Application runs in multiple runtimes spread across multiple hosts (VIII)

Immutable deployment

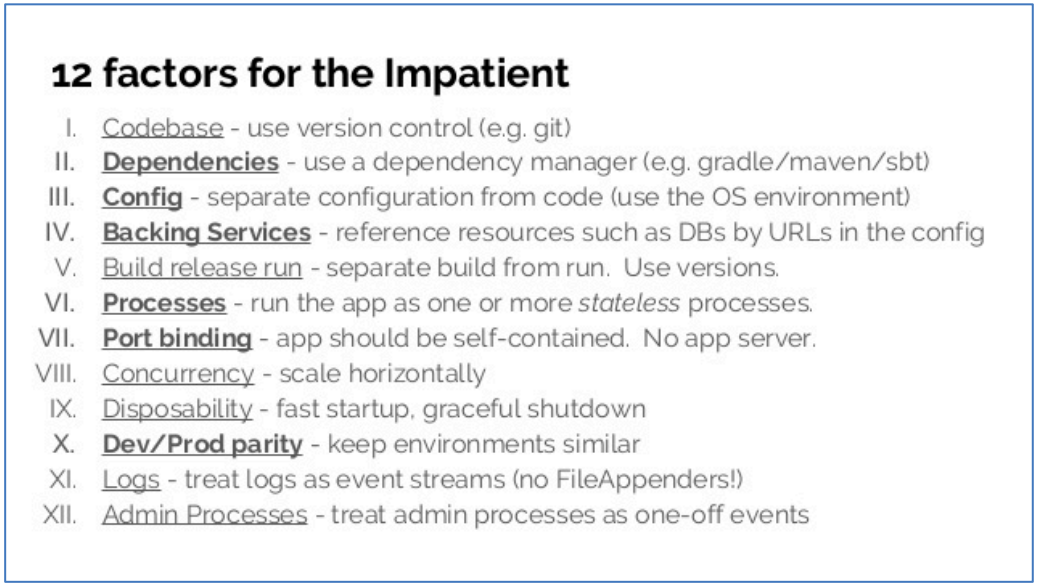
- A runtime is not patched, it's replaced (IX)
- A runtime is stateless (VI)
- Shared functionality in backing services (IV)

Elasticity

- Automatic scale-out and scale-in to maintain performance
- Achieved via containerization

Pay-as-you-go charging model

- Pay for what you use



https://12factor.net

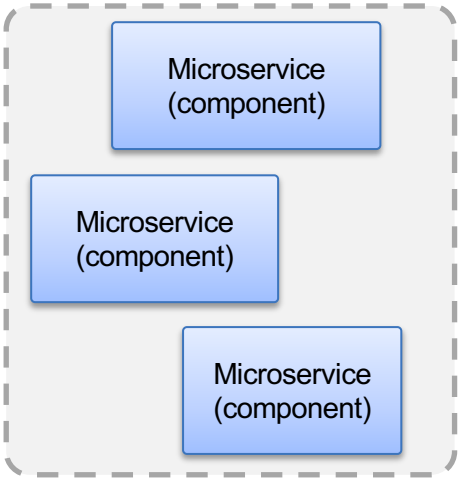
Microservices: Making developers more efficient

- An **engineering approach** that reduces an application into **single-function** modules
- They have well-defined **interfaces** that are independently deployed
- They are operated by a **small team** which owns the entire **lifecycle** of the service
- Microservices accelerate delivery by
  - Minimizing communication and coordination between people
  - Reducing the scope and risk of change

Microservices Architecture ?

Simplistically, microservices architecture is about breaking down large silo applications into more manageable fully decoupled pieces

Monolithic application



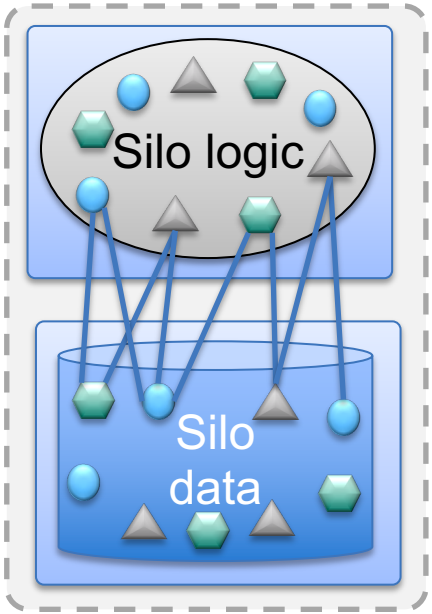
Microservices application

Agility
Scalability
Resilience

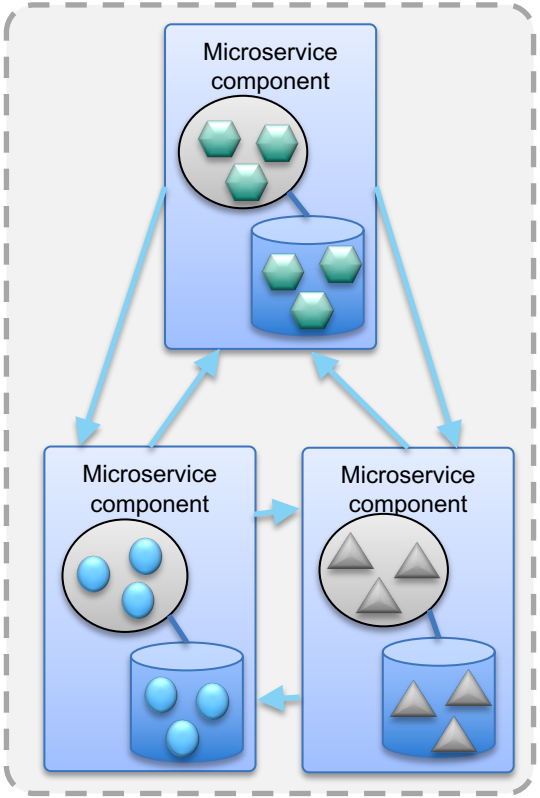
A *microservice* is a granular <u>decoupled</u> component within a broader application

With microservices, encapsulation is key

Related logic and data should remain together, and which means drawing strong boundaries between microservices.



Monolithic application



Microservices application

Example operating system boundaries

Why Microservices?

Small scoped, independent, scalable components

Scaling

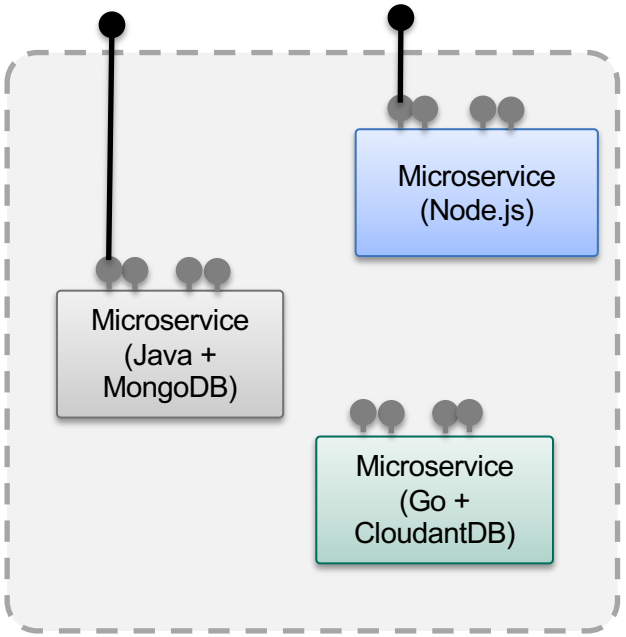
- Elastic scalability
- Workload orchestration

Agility

- Faster iteration cycles
- Bounded context (code and data)

Resilience

- Reduced dependencies
- Fail fast



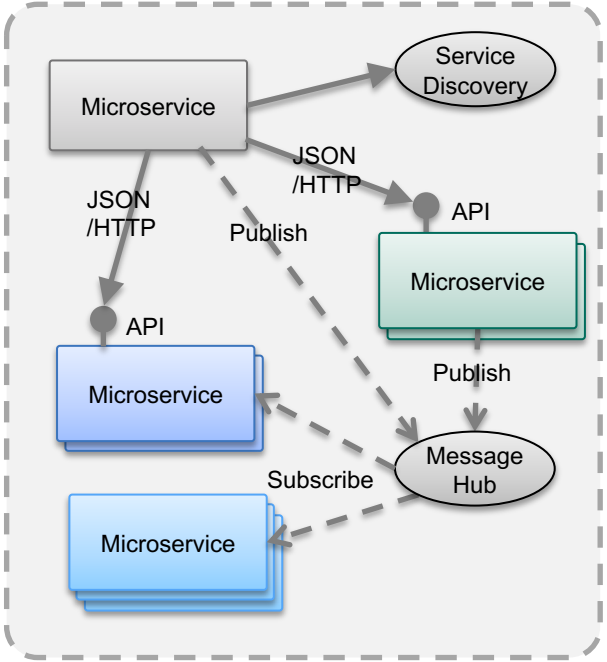
Microservices application

Microservices inter-communication

Aim is decoupling for robustness

Compose a complex application using
- “small”
- independent (autonomous)
- replaceable
- processes

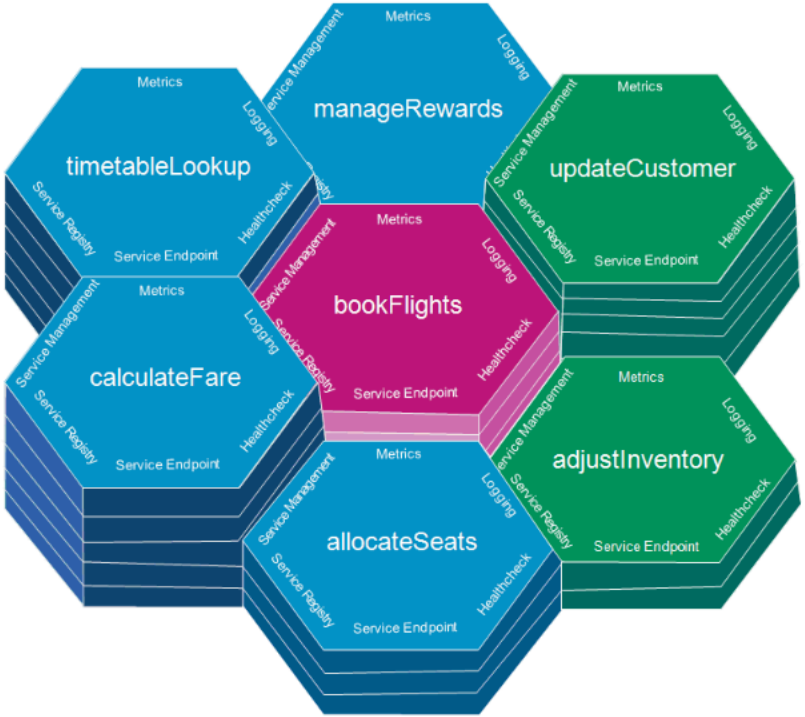
that communicate
- via language-agnostic APIs
- synchronously and asynchronously



Microservices application

Sample application that uses microservices

- In this Airline reservation application, each service includes
  - Logging
  - Metrics
  - Health check
  - Service endpoint
  - Service registry
  - Service management
- Different Languages
- Sharing achieved through library sharing.
- Different number of instances



Advantages / Challenges of Microservices

Advantages

- Developed independently
- Developed by a single team
- Developed on its own timetable
- Each can be developed in a different language
- Manages its own data
- Scales and fails independently

Challenges

- Greater operational complexity
- Developers must have significant operational skills (DevOps)
- Service interfaces and versions
- Duplication of effort across service implementations
- Extra complexity of creating a distributed system:
- Designing decoupled, non-transactional systems is difficult
- Locating service instances
- Maintaining availability and consistency with partitioned data
- End-to-end testing

Comparing monolithic and microservices architectures

Category	Monolithic architecture	Microservices architecture
Architecture	Built as a single logical executable	Built as a suite of small services
Modularity	Based on language features	Based on business capabilities
Agility	Changes involve building and deploying a new version of the entire application	Changes can be applied to each service independently
Scaling	Entire application scaled when only one part is the bottleneck	Each service scaled independently when needed
Implementation	Typically entirely developed in one programming language	Each service can be developed in a different programming language
Maintainability	Large code base is intimidating to new developers	Smaller code bases easier to manage
Deployment	Complex deployments with maintenance windows and scheduled downtimes	Simple deployment as each service can be deployed individually, with minimal downtime

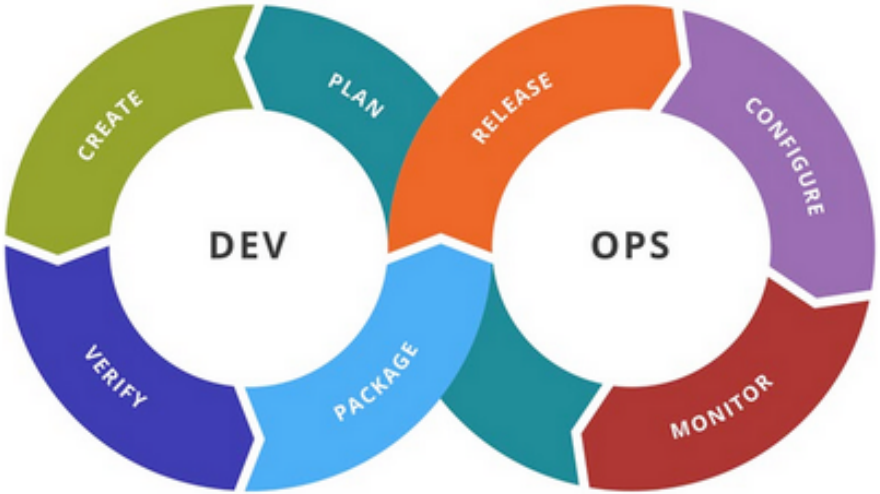
DevOps

DevOps is an approach that combines 2 major trends:

1. Agile infrastructure
2. Collaboration between development and operations

DevOps adoption drivers:

1. Agility &Speed
2. Security
3. Cohesion



.. the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.

Continuous Integration

Continuous integration is a software development practice where numbers of a team **integrate their work frequently,** usually each person integrates at least daily – leading to multiple integrations per day.

Each integration is **verified** by an automated build (including test) to detect errors as quickly as possible.

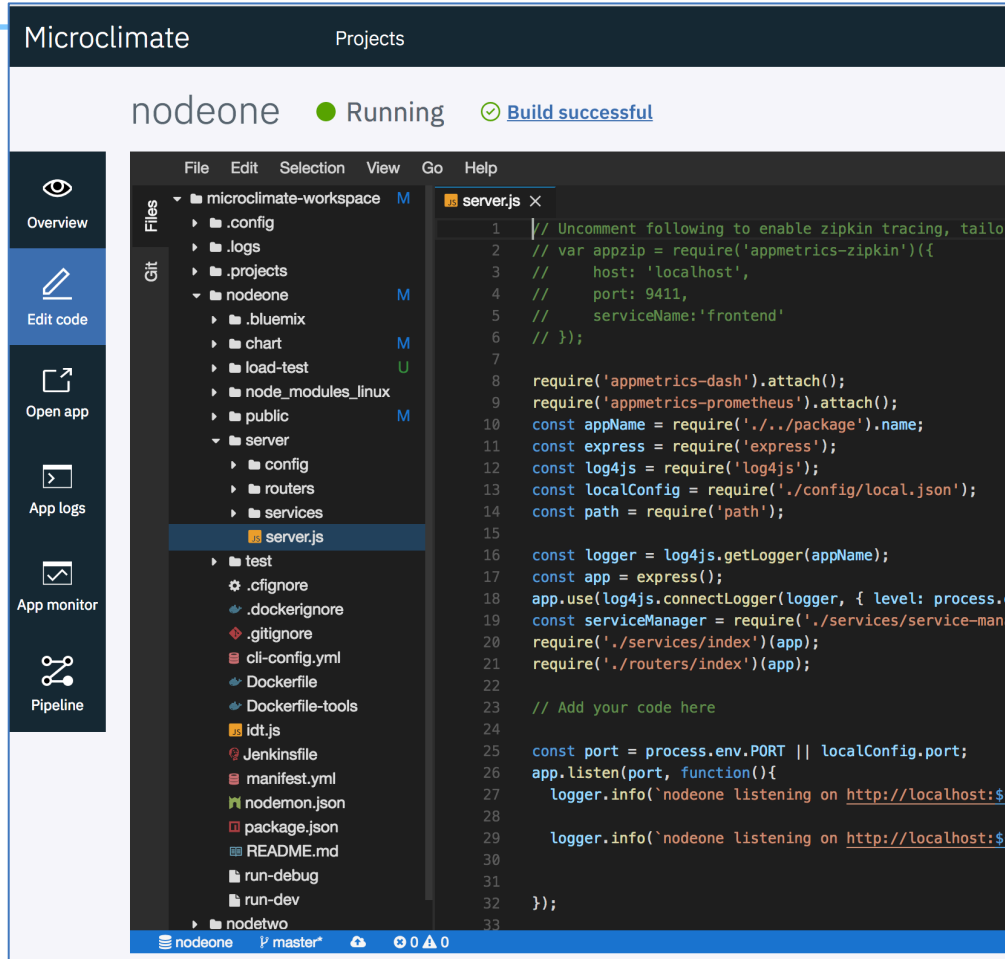
Martin Fowler

http://martinfowler.com/articles/continuousIntegration.html

What is Microclimate ?

Helping developers build services for IBM Cloud Private

- Service-oriented and scalable
- Unified development environment
- Cloud technology
- Built using the latest standards
- End-to-end Docker deployment
- Fast, flexible development
- Multiple IDE options
- Integrated DevOps PipeLine

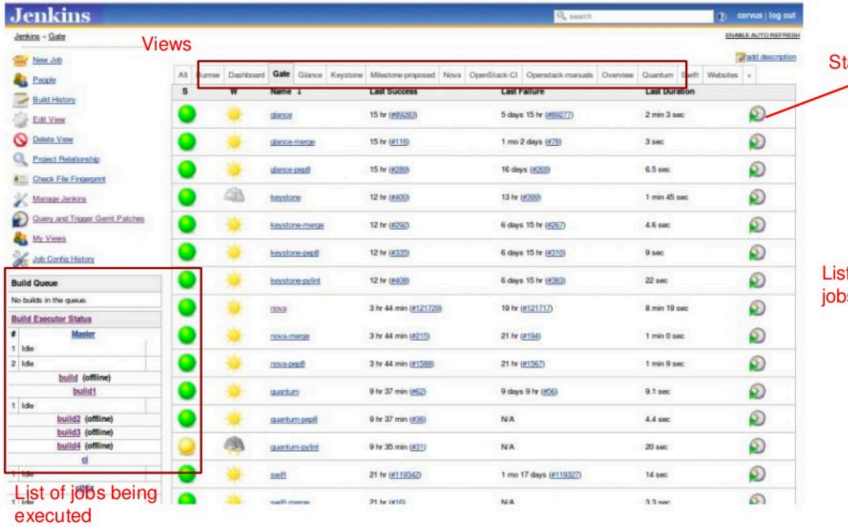


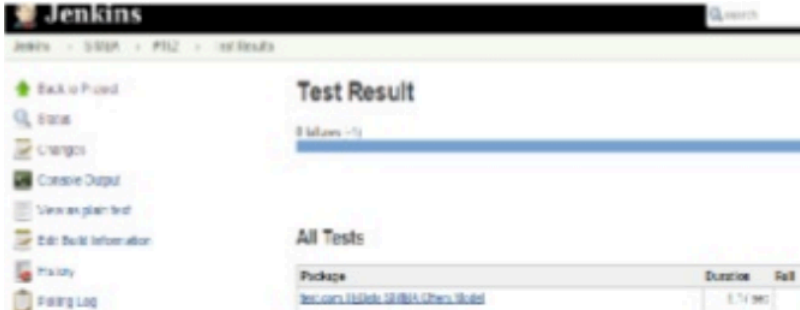
Benefits of using **Microclimate**

- Accelerated software delivery, from weeks to days to minutes, leveraging a Continuous Delivery **Pipeline**
- Simple and intuitive **user experience**, from development to production
- Support for **rapid** application development and testing cycles with greater agility, scalability, and security
- **Reduced costs and complexity** with seamless portability across popular cloud providers including public, dedicated, private, and hybrid.
- **Real-time diagnosis** and resolution of app infrastructure, minimizing downtime and maintaining SLAs
- **Easy connection** between existing applications to new cloud services (like Watson cognitive services) to discover actionable insights

What is **Jenkins?**

- Jenkins
  - Popular open-source framework for Continuous Integration
  - Monitors execution of repeated jobs; examples: cron jobs or builds
  - Generally used for:
    - Building/testing software projects continuously
    - Monitoring executions of externally-run jobs, such as cron jobs





Continuous Delivery, Continuous Deployment

Continuous delivery is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.

Continuous deployment is a similar approach that aims at automating **deployments** rather than having manual processes.

What is UrbanCode Deploy?

- **What**: the deployable items - binary files, static content, MW updates, DB changes and configs
- **How**: by combining deployable items with processes to create components and designing applications that coordinate and orchestrate multicomponent deployments.
- **Where:** the targeted hosts and environments

Links about Microservices

- Building Microservices (O'REILLY)

http://shop.oreilly.com/product/0636920033158.do

https://martinfowler.com/microservices/

- Microservices Architecture (IBM)

https://developer.ibm.com/architecture/microservices

https://www.ibm.com/cloud/garage/architectures/microservices/2_1

http://www.redbooks.ibm.com/abstracts/sg248275.html?Open

http://www.redbooks.ibm.com/redpapers/pdfs/redp5271.pdf

- IBM Code (IBM)

https://developer.ibm.com/code/technologies/microservices/