Criteria for Choosing a Technology for Microservices

With microservices, you can build a reliable platform to support continuous innovation with business while taking advantage of diversity in languages. It is acceptable to say that microservices has benefits over monolith architecture but sometimes it adds additional operational overheads that’s why we no need add multiple languages also to be part of that operational overhead. It is important to standardize your microservices technology stack by choosing the programming language based on your business needs. Given below are some criteria to evaluate the programming language for microservices development:

* Culture of automation
* Highly observable
* Support for automation
* Consumer-first approach
* Independent deployment
* Modelled around business domain
* Decentralization of components
* Support for continuous integration

## Best Languages for Microservices

Let us explore the languages that support microservices development in detail:

### **1. Java**

It is all about annotation what makes java an easy approach for microservices development. This feature makes Java Microservices much easier to develop when powered by Microservices frameworks. It offers more value in readability and code reduction with readymade configuration libraries. It also offers a user Interface, model components as well as connectivity to back-end resources, everything within the boundaries of a single, isolated and independently deployed apps.

Further many of Java EE standards are well suited for microservices applications like:

* JAX-RS for APIs
* JPA for data handling
* CDI for dependency injection & lifecycle management

In addition, service discovery solutions like Consul, Eureka from Netflix or Amalgam8 are effortless in connecting with Java Microservices.

Also, there are Frameworks for developing Microservices architecture.

* **Spring Boot** – This framework works on top of various languages for Aspect-Oriented programming, Inversion of Control and others
* **Drop wizard** – This Java microservices framework assembles stable and mature libraries of Java into a simple and light-weight package
* **Restlet** – It supports developers to build better web APIs, which trail the REST architecture model
* **Spark** – One of the best Java Microservices frameworks, supports creating web apps in Java 8 and Kotlin with less effort

### **2. Golang**

For an enhancement of the existing project, the Golang can be a good choice for microservices development. Golang, also known as Go is popular for its concurrency and API support in terms of microservices architecture. With the Golang’s concurrency possibility, you can expect increased productivity of various machines and cores. It includes a powerful standard for developing web services. It is exclusively designed for creating large and complex applications. It has two impressive frameworks for microservices development:

* **GoMicro** – It is an RPC framework, which comes with the advantages like Load balancing, server packages, PRC client, and message encoding.
* **Go Kit** – The key difference of Go Kit from GoMirco is it needs to be imported into a binary package. Moreover, it is advanced for explicit dependencies, Domain-driven design, and declarative aspect compositions.

In addition to simple syntax, Go microservices architecture includes excellent testing support as it makes it simple to write robust tests as well as embed them flawlessly into workflows.

### **3. Python**

It is a high-level programming language that offers active support for integration with various technologies. Prototyping in Python is faster and easier when compared to other frameworks and languages. It includes powerful substitutes for heavy implementations like Django. Microservices Python ensures compatibility with legacy languages like ASP and PHP, which allows you to create web service front-ends to host Microservices.

With all these benefits, Microservices Python is considered to have an edge over other languages. Developers who implement Microservices Python use a RESTful API approach - a comprehensive way of utilizing web protocols & software to remotely manipulate objects. Monitoring becomes easy since it is now broken into components. Given below is a broad range of Python microservices frameworks to choose from for your web application development. Some of them are as follows:

* **Flask** – Most popular Python Micro framework based on Jinja2 and Werkzeug

has great options and can easily be used to create other HTTP-based microservices

* **Falcom** – Create smart proxies, cloud APIs and app back-ends
* **Bottle** – Simple, lightweight and fast WSGI micro framework
* **Nameko**– Best among the Python Microservices frameworks that allows developers to concentrate on application logic
* **CherryPy** – Mature, Python object-oriented web framework

### **4. .Net**

The .NET platform was designed to reduce programming errors and increase productivity by using a modular approach to software design. ASP.Net, the .Net framework for web development makes it simple to build the APIs that becomes the microservices. It includes built-in support for building and deploying microservices using Docker containers. .Net comes with APIs that can simply consume microservices from any application you developed across multiple platforms including desktop, mobile, web, gaming and more. If you have an application, you can start adopting .Net microservices without entirely revamping that application. Premeditated .Net Docker images have already been done and available on Docker Hub, helping you to concentrate only on building your microservices.

The .Net microservices architecture allows mixing of technologies. Such as you can mix the .Net microservices with applications written in Java, Node JS, or any other languages. This allows a gradual migration to .Net core technology for new microservices that function in combination with other microservices and with services built with other technologies. Also, the .Net microservices can run on all leading cloud platforms.

The future of microservices leads us closer to serverless architecture, Event driven, function based. Microservices is an extensive concept that applies to churn apps, products or solutions to more granular and modular level. Keep in mind, it is not easy to with t microservices architecture from scratch but if you do so the boundaries should be clearly defined at the business requirement level.

### **5. Node JS**

Node JS became the go-to platform in the past few years for enterprises and startups who want to embrace microservices. Node JS is built with the Google’s V8 Java Script runtime engine. A rich database of multiple JavaScript modules provided by Node.js simplifies the application development at a great scale. Software architects [prefer Node.js](https://www.cuelogic.com/blog/why-node-the-fundamental-difference-between-node-and-other-languages) as a technology partner for developing [JSON](http://www.json.org/) API-based applications, I/O bound applications, data streaming applications, single-page applications, and data-intensive real-time applications.

**Key benefits of Node.js**

* **Single-threaded**: With event looping, the server uses a non-blocking mechanism to respond.
* **Super-fast**: Codes are executed quickly on Google’s [V8 JavaScript Engine](https://v8.dev/)
* **Event-driven**: ‘Events of Node.js’ is a notification system that enables the application server to capture the response of the previous API call.
* **Buffer-less**: There is no buffering as data is simply released in chunks.
* **Asynchronous**: The non-blocking, non-synchronous Node.js libraries move to the next API and do not await the return data of the previous API.
* **Highly-scalable**: Servers can handle as many requests as coming their way

Microsoft, PayPal, Uber, eBay, and Yammer using [Node.js application development](https://www.cuelogic.com/nodejs-development) for their projects.