Menurut Paulo Merson (Software Architect at TCU; SOA/microservices trainer and consultant)

Brazilian Federal Court of Accounts (TCU) Carnegie Mellon University

1. Deployability: Lebih gesit untuk meluncurkan versi terbaru karena siklus build + test + build yang lebih pendek. Juga, fleksibilitas untuk menggunakan layanan keamanan, replikasi, persistensi, dan konfigurasi pemantauan
2. Reliability: Kesalahan microservice hanya berpengaruh pada mikroservice itu sendiri dan konsumennya, sedangkan dalam model monolitik kesalahan layanan dapat merusak seluruh monolit.
3. Availability: Mengeluarkan versi baru dari microservice hanya memerlukan sedikit *downtime*, sedangkan meluncurkan versi baru dari layanan di monolith membutuhkan restart yang lebih lambat dari keseluruhan monolit.
4. Scalability: setiap mikroservice dapat diskalakan secara terpisah menggunakan pool, cluster, grid. Karakteristik penyebaran membuat microservices sangat cocok untuk elastisitas pada *cloud*.
5. Modifiability: lebih fleksibel untuk menggunakan framework baru, libraries, datasources, dan *resources* lain. Microservice juga loosely-coupled, modular components hanya dapat diakses oleh contracts (yg berhak mengakses), dan cenderung tidak mudah berubah menjadi big ball of mud (istilah apabila software sudah sangat besar dan sulit ditangani).
6. Management: upaya pengembangan aplikasi dibagi di antara tim yang lebih kecil dan bekerja lebih mandiri.
7. Design autonomy: tim memiliki kebebasan untuk menggunakan teknologi, kerangka kerja, dan pola yang berbeda untuk merancang dan mengimplementasikan setiap microservice, dan dapat mengubah dan memindahkan masing-masing microservice secara mandiri

Atau dapat disingkat menjadi 4 point

1. Agility: Dengan memecah fungsionalitas ke tingkat yang paling dasar dan menghubungkannya dengan service lain, *developer* dapat fokus hanya memperbarui potongan aplikasi yang relevan. Ini menghilangkan proses integrasi yang biasanya ada pada aplikasi monolitikmicroservice mempercepat proses development, mengubahnya menjadi proses yang dapat diselesaikan dalam hitungan minggu dan bukan bulan.
2. Efficiency: Memanfaatkan arsitektur berbasis microservices dapat menghasilkan penggunaan kode dan infrastruktur yang jauh lebih efisien. Tidak jarang mengalami penghematan biaya sampai dengan 50% dengan mengurangi jumlah infrastruktur yang diperlukan untuk menjalankan aplikasi tertentu.
3. Resiliency: Dengan menyebar fungsi di beberapa layanan menghilangkan kerentanan aplikasi ke satu titik kegagalan. Menghasilkan aplikasi yang dapat berkinerja lebih baik, mengalami lebih sedikit *downtime*.
4. Revenue: Iterasi yang lebih cepat dan downtime yang menurun dapat meningkatkan pendapatan (baik menggunakan efisiensi yang dihasilkan melalui ideologi *chargeback*, atau dengan meningkatkan keterlibatan pengguna).