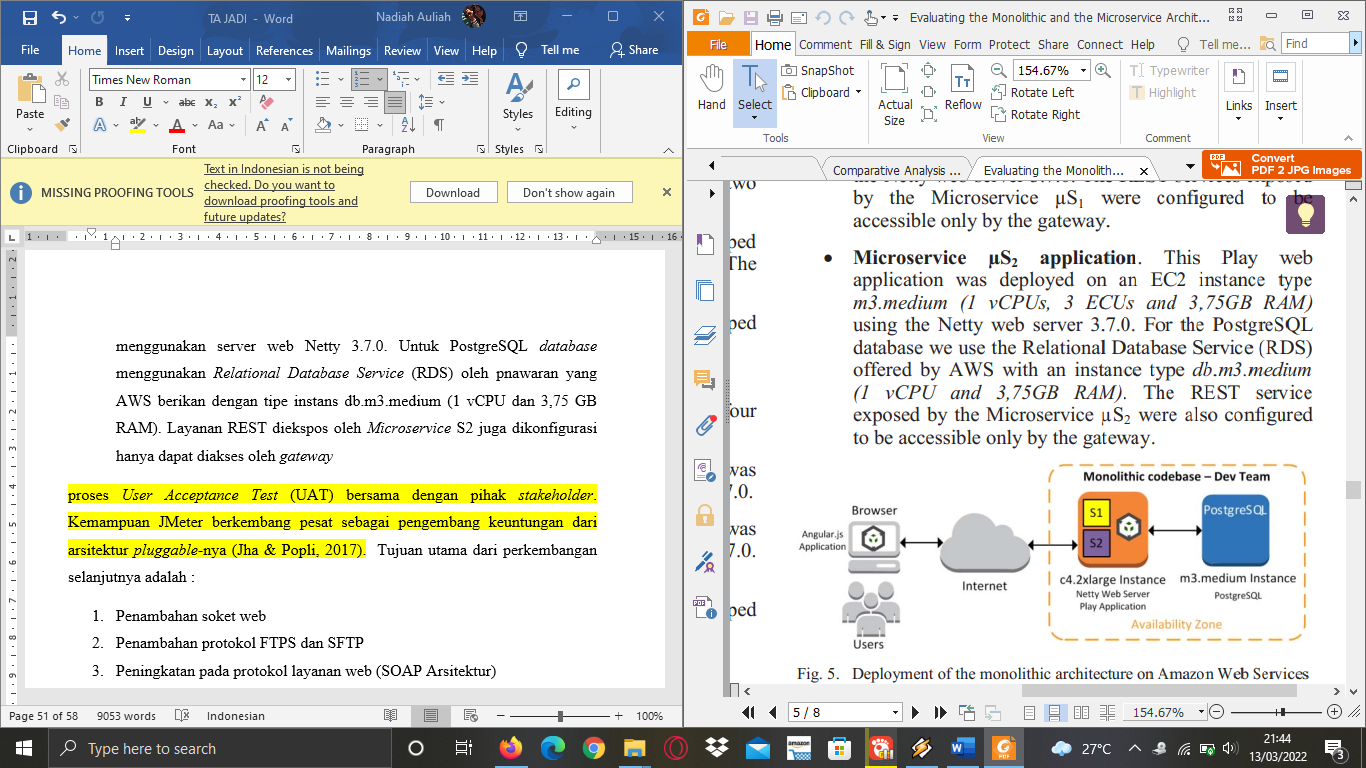
Pada tahap *system testing*, dilakukan berbagai macam pengujian keseluruhan fitur menggunakan pengujian *Jmeter Apache testing*. Pengujian dilakukan untuk mengetahui kesalahan atau *error* yang terjadi pada sistem, untuk melakukan *testing* dilakukan pengujian pada aplikasi JMeter. Sebelum melakukan testing menggunakan JMeter, *developer* harus mengetahui karakteristik dari metode atau konsep metode *Microservice* sehingga mudah melakukan uji coba pada JMeter. Konsep pengukuran *microservice* memiliki kriteria sebagai berikut :

1. **Aplikasi Layanan Mikro μS1.** Aplikasi Web Play ini mengarah pada jenis instans EC2 c4.xlarge (4 vCPU, 16 ECU dan RAM 7,5 GB) menggunakan *server* web Netty 3.7.0. Layanan REST terbuka oleh *Microservice* S1 dikonfigurasi menjadi hanya dapat diakses oleh *gateway.*
2. **X Aplikasi Microservice S2**. Web Play ini aplikasi mengacu pada jenis instans EC2 m3.medium (1 vCPU, 3 ECU, dan 3,75 GB RAM) menggunakan server web Netty 3.7.0. Untuk PostgreSQL *database* menggunakan *Relational Database Service* (RDS) oleh pnawaran yang AWS berikan dengan tipe instans db.m3.medium (1 vCPU dan 3,75 GB RAM). Layanan REST diekspos oleh *Microservice* S2 juga dikonfigurasi hanya dapat diakses oleh *gateway*



**Gambar 3.5** Perbandingan dengan arsitektur *monolithic*.

Arsitektur mobolitik memerlukan jaringan internet yang cukup kuat dikarenakan memiliki pengendali sistem hanya 1 yakni server utama.

1. ***Gateway Aplication***. Merupakan aplikasi *web Play* yang di-*deploy* pada tipe instans EC2 m3.medium (1 vCPU, 3 ECU, dan RAM 3,75 GB) menggunakan web Netty server 3.7.0. Layanan REST dari gateway adalah terpapar internet.
2. ***Front-end Aplication***. Mirip dengan monolitik aplikasi, file statis dari aplikasi Angular.js disimpan di *Gateway*. Saat pengguna masuk ke aplikasi *web*, dalam permintaan pertama aset *Angular.js* diunduh ke *browser* dari gerbang; maka layanan REST diekspos oleh *gateway* dikonsumsi dari aplikasi Angular.js (dijalankan di b*rowser*) menggunakan JSON

Dalam pengimplementasiannya, monolitik hanya memiliki dua macam arsitektur. Sedangkan untuk microservice, metode ini memiliki tiga macam arsitektur yang diurutkan berdasarkan level. Berikut adalah arsitektur monolitik da *microservice :*

* *Monolitthic*

1. ***Web Aplication***. Merupakan aplikasi yang dikembangkan menggunakan *Play* 2.2.2, Skala 2.10.2 dan Java 1.7.0. *Database* relasional yang digunakan adalah PostgreSQL 9.3.6.
2. ***Front-End Aplication***. Aplikasi ini dikembangkan menggunakan Angular.js 1.3.14 (HTML5, CSS, dan jQuery).

* *Microservice*

1. ***Microservice μS1 Application***. Aplikasi ini adalah dikembangkan menggunakan *Play* 2.2.2, *Scala* 2.10.2 dan Java 1.7.0.
2. ***Microservice μS2 Application***. Aplikasi ini adalah dikembangkan menggunakan *Play* 2.2.2, *Scala* 2.10.2 dan Java 1.7.0. Basis data relasional yang digunakan adalah PostgreSQL 9.3.6.
3. **Gateway application**. Aplikasi ini dikembangkan menggunakan Play 2.2.2, Scala 2.10.2 dan Java 1.7.0.

Jenis instansi yang digunakan pada kedua arsitektur terpilih berdasarkan tes kinerja yang terjadi. Untuk setiap arsitektur kami memilih contoh yang lebih kecil yang mendukung persyaratan bisnis (keluarga tipe instansi t2 tidak diperhitungkan karena volatilitas CPU-nya). Untuk melakukan proses validasi keberhasilan sistem, dirancang testing pada stakeholder menggunakan Teknik UAT atau *User Acceptance Test* (UAT) yang akan dijelaskan lebih lanjut pada tahap *retrospective*. Adapun arsitektur susunan testing menggunakan JMeter adalah sebagai berikut :



**Gambar 3. 6** Arsitektur JMeter Apache

Gambar 3.5 menjelaskan proses *testing* pada sistem menggunakan aplikasi JMater Apache. Proses bermula dari *user* yang meminta koneksi pada server untuk masuk kedalam jaringan sistem. Setelah permintaan dikirimkan, server merespon, merekam jejak informasi lalu mendapatkan hasil data untuk mengubahnya menjadi informasi sementara. Data statis sementara akan disimpan oleh server untuk ditampilkan kepada *user* dan diakhiri oleh proses selesai. Untuk memperjelas proses, proses dikategorikan dalam beberapa *statement* sebagai berikut :

* *Peformance Testing*

1. ***Load Testing***

Hal ini dilakukan untuk memeriksa apakah server dapat memenuhi spesifikasi atau tidak. Ini analog dengan pengujian volume dan menentukan bagaimana aplikasi menangani tugas-tugas besar.

1. ***Stress Testing***

Pengujian Stres Ini memeriksa perilaku aplikasi di bawah semburan puncak aktivitas. Hal ini dilakukan untuk mengevaluasi sistem atau komponen pada atau melampaui batas persyaratan yang ditentukan. sudah selesai untuk menemukan titik akhir di mana *server* mulai berperilaku tidak normal atau mulai menerjang, titik ini disebut titik stres atau titik lutut atau titik jatuh. Misalnya, jika ada penjualan atau penawaran di situs web pada hari tertentu, lalu berapa banyak pengguna yang dapat menangani secara konsisten.

1. ***Spike Testing***

Pengujian lonjakan dilakukan untuk menguji fluktuasi selama periode waktu dan kemudian mencari tahu alasan mereka. Jenis kinerja ini pengujian dilakukan untuk memverifikasi stabilitas sistem selama ledakan beberapa pengguna pada saat yang sama atau beban yang bervariasi selama periode waktu yang bervariasi.

1. ***Capacity Testing***

Pengujian Kapasitas Ini mengukur kapasitas keseluruhan sistem dan menentukan pada titik apa waktu respons dan throughput menjadi tidak dapat diterima. Pengujian kapasitas dilakukan dengan beban normal untuk tentukan kapasitas ekstra di mana kapasitas tegangan adalah ditentukan dengan membebani sistem hingga gagal, yaitu juga disebut beban tegangan untuk menentukan kapasitas maksimum sebuah sistem. Hal ini dilakukan untuk memastikan skalabilitas dan stabilitas untuk tujuan masa depan.

* Performance Testing goals

1. ***Stability***

Uji stabilitas dirancang untuk menentukan apakah web aplikasi akan tetap dapat digunakan selama rentang waktu yang diperpanjang. Hal ini dicapai melalui pengujian rendam. Server harus bekerja secara konsisten. Misalnya, untuk beban konstan 1000 pengguna per kedua selama 15 hari dengan lingkungan dan konfigurasi yang sama, waktu respons server tidak boleh berfluktuasi sejak dimuat belum diubah.

1. ***Response Time***

Waktu respons Response Time adalah jeda waktu antara pengguna yang mengirim meminta dan menerima tanggapan mereka. Ini mewakili berapa lama seorang pengguna sedang menunggu permintaannya untuk diproses oleh aplikasi. Dua faktor utama membuat waktu download lambat: Ukuran halaman web besar dan kinerja server lambat. itu semua tentang mendefinisikan pengukuran. Tes kinerja selesai untuk meningkatkan kinerja tetapi untuk mengetahui hal ini, arus dasar harus diketahui. Artinya seberapa cepat aplikasinya harus memenuhi spesifikasi

1. ***Scalability***

Waktu Transaksi adalah penentu utama Skalabilitas. Waktu Transaksi tidak berarti skalabilitas yang baik – sebenarnya ini adalah antitesis itu. Skalabilitas ditentukan oleh jumlah klien simultan dilayani selama periode waktu tertentu dengan waktu tunda yang terbatas.

* Skala Pengujian

Pengujian kinerja adalah pengujian yang dilakukan untuk menentukan seberapa cepat beberapa aspek sistem bekerja di bawah a beban kerja tertentu. Dengan kata lain, untuk memeriksa fungsionalitas di bawah beban disebut pengujian kinerja. Tujuh kegiatan pengujian kinerja inti:

1. Identifikasi Lingkungan Pengujian
2. Identifikasi Kriteria Penerimaan Kinerja
3. Rencana dan Tes Desain
4. Konfigurasikan Lingkungan Uji
5. Menerapkan Desain Uji
6. Jalankan Tes
7. Analisis, Laporkan, dan Uji Ulang

Ketujuh kegiatan inti ini tidak dengan sendirinya membentuk pendekatan untuk pengujian kinerja; sebaliknya, mereka mewakili landasan di mana suatu pendekatan dapat dibangun yaitu tepat untuk proyek tersebut.

Secara umum masuk akal untuk mulai mengidentifikasi, atau setidaknya memperkirakan, karakteristik kinerja yang diinginkan dari aplikasi di awal siklus hidup pengembangan. Ini bisa jadi dicapai paling sederhana dengan mencatat kinerja karakteristik yang disamakan oleh pengguna dan pemangku kepentingan Anda Penampilan yang bagus. Catatan dapat diukur di lain waktu. Kelas karakteristik yang sering berkorelasi dengan pengguna atau kepuasan pemangku kepentingan biasanya meliputi:

1. *Response time*: Waktu yang dibutuhkan oleh *server* untuk memproses satu transaksi. Misalnya, produk katalog harus ditampilkan dalam waktu kurang dari tiga detik.
2. Latensi: Waktu yang diperlukan untuk memproses permintaan, seperti dirasakan oleh pengguna. Latensi termasuk penundaan jaringan serta waktu respon.
3. *Throughput*: Jumlah transaksi yang diproses per satuan waktu. Misalnya, sistem harus mendukung 25 pesanan buku per detik.
4. *Workload*: Jumlah transaksi yang harus diproses per satuan waktu. Misalnya, server menerima 500 transaksi per menit, tetapi mampu memproses hanya 80% dari beban itu.

Beban kerja=500 transaksi per menit

Throughput=400 transaksi per menit

Idealnya throughput = beban kerja

Ketika beban kerja melebihi kapasitas, throughput < beban kerja dan transaksi tertunda atau hilang.

1. Pemanfaatan Sumber Daya: Misalnya, prosesor pemanfaatannya tidak lebih dari 75 persen. Lainnya sumber daya penting yang perlu dipertimbangkan untuk tujuan pengaturan adalah memori, input/output disk (I/O), dan I/O jaringan.
2. Standar Deviasi: Menurut definisi, satu standar deviasi adalah jumlah varians dalam kumpulan data..

Ketika proses selesai, *user* mendapatkan data yang diperlukan. Jika tidak terdapat masalah, maka proses berlanjut pada tahap *retrospective*.