AWSWhitepaper

Dasar-dasar AWS Multi-Wilayah

Dasar-dasar AWS Multi-Wilayah: AWSWhitepaper

Table of Contents

Dasar-dasar AWS Multi-Wilayah

Tanggal publikasi: 20 Desember 2022 (Revisi dokumen)

Abstrak

Paper tingkat 300 yang canggih ini ditujukan untuk arsitek cloud dan pemimpin senior yang membangun beban kerja tentang AWS siapa yang tertarik menggunakan arsitektur Multi-wilayah untuk meningkatkan ketahanan beban kerja mereka. Paper ini mengasumsikan pengetahuan dasar AWS infrastruktur dan layanan. Ini menguraikan kasus penggunaan Multi-wilayah yang umum, berbagi konsep dan implikasi Multi-wilayah mendasar seputar desain, pengembangan, dan penerapan, dan memberikan panduan preskriptif untuk membantu Anda menentukan dengan lebih baik apakah arsitektur Multi-wilayah tepat untuk beban kerja Anda.

Apakah Anda Well-Architected?

AWSWell-Architected Framework membantu Anda memahami pro dan kontra dari keputusan yang Anda buat saat membangun sistem di cloud. Enam pilar Kerangka memungkinkan Anda mempelajari praktik terbaik arsitektur untuk merancang dan mengoperasikan sistem yang andal, aman, efisien, hemat biaya, dan berkelanjutan. Dengan menggunakan AWS Well-Architected Tool, tersedia tanpa biaya di AWS Management Console, Anda dapat meninjau beban kerja Anda terhadap praktik terbaik ini dengan menjawab serangkaian pertanyaan untuk setiap pilar.

Untuk panduan ahli dan praktik terbaik lainnya untuk arsitektur cloud Anda—penerapan arsitektur referensi, diagram, dan whitepaperer—lihat Pusat Arsitektur. AWS

Pengantar

Masing-masing Wilayah AWSterdiri dari beberapa Availability Zone yang independen dan terpisah secara fisik dalam suatu wilayah geografis. Pemisahan logis yang ketat antara layanan perangkat lunak di setiap Wilayah dipertahankan. Desain yang bertujuan ini memastikan bahwa kegagalan infrastruktur atau layanan di satu Wilayah tidak akan mengakibatkan kegagalan yang berkorelasi di Wilayah lain.

Sebagian besar AWS pelanggan dapat mencapai tujuan ketahanan mereka untuk beban kerja di satu Wilayah menggunakan beberapa Availability Zone (AZ) atau layanan Regional. AWS Namun, sebagian pelanggan mengejar arsitektur Multi-wilayah karena tiga alasan.

- Mereka memiliki ketersediaan tinggi dan kontinuitas persyaratan operasi untuk beban kerja tingkat tertinggi mereka yang mereka yakini tidak dapat dipenuhi di satu Wilayah.
- Mereka perlu memenuhi persyaratan kedaulatan data (seperti kepatuhan terhadap hukum, peraturan, dan kepatuhan setempat) yang memerlukan beban kerja untuk beroperasi dalam yurisdiksi tertentu.
- Mereka perlu meningkatkan kinerja dan pengalaman pelanggan untuk beban kerja dengan menjalankan beban kerja di lokasi yang paling dekat dengan pengguna akhir.

Paper ini berfokus pada ketersediaan tinggi dan kontinuitas persyaratan operasi, dan membantu Anda menavigasi pertimbangan untuk mengadopsi arsitektur Multi-wilayah untuk beban kerja. Kami menjelaskan konsep dasar yang berlaku untuk desain, pengembangan, dan penyebaran beban kerja Multi-wilayah, bersama dengan kerangka kerja preskriptif untuk membantu Anda menentukan apakah arsitektur Multi-wilayah adalah pilihan yang tepat untuk beban kerja tertentu. Anda perlu memastikan arsitektur Multi-region adalah pilihan yang tepat untuk beban kerja Anda, karena arsitektur ini menantang, dan ada kemungkinan bahwa, jika tidak dilakukan dengan benar, ketersediaan keseluruhan beban kerja dapat berkurang.

Rekayasa dan operasi untuk ketahanan di satu Wilayah

Sebelum menyelami konsep Multi-wilayah, mulailah dengan mengonfirmasi bahwa beban kerja Anda sudah sekuat mungkin di satu Wilayah. Untuk mencapai hal ini, evaluasi beban kerja Anda terhadap Pilar Keandalan dan Pilar Keunggulan Operasional dari Kerangka AWS Well-Architected, dan buat perubahan yang diperlukan untuk mengadopsi praktik terbaik yang direkomendasikan. Konsep-konsep berikut tercakup dalam AWS Well-Architected Framework:

- Segmentasi beban kerja berdasarkan batas domain
- Kontrak layanan yang terdefinisi dengan baik
- Manajemen ketergantungan dan kopling
- Menangani kegagalan, percobaan ulang, dan strategi mundur
- Operasi idempoten dan transaksi stateful vs stateless
- Kesiapan operasional dan manajemen perubahan
- Memahami kesehatan beban kerja
- Menanggapi peristiwa

Untuk meningkatkan ketahanan Wilayah tunggal, tinjau dan terapkan konsep yang dibahas dalam pola ketahanan Multi-AZ Tingkat Lanjut untuk menangani kegagalan abu-abu. Paper ini memberikan praktik terbaik seputar penggunaan replika di setiap Availability Zone untuk menahan kegagalan, dan memperluas konsep Multi-AZ yang diperkenalkan di AWS Well Architected. Setelah Anda sepenuhnya menerapkan konsep yang direkomendasikan dan praktik terbaik untuk mencapai ketahanan tertinggi di satu Wilayah, beban kerja tertentu dapat dievaluasi terhadap dasar-dasar arsitektur Multi-wilayah untuk menentukan apakah ketahanan beban kerja dapat ditingkatkan menggunakan pendekatan Multi-wilayah.

Multi-Region fundamental 1: Memahami persyaratan

Seperti disebutkan sebelumnya, ketersediaan dan kontinuitas operasi yang tinggi adalah alasan umum untuk mengejar arsitektur Multi-wilayah. Metrik ketersediaan mengukur persentase waktu beban kerja tersedia untuk digunakan selama periode tertentu, sementara metrik kontinuitas operasi mengukur pemulihan untuk kejadian berskala besar dan biasanya berdurasi lebih lama.

Mengukur ketersediaan adalah proses yang hampir berkelanjutan. Pengukuran atau metrik spesifik dapat bervariasi, tetapi biasanya menyatu di sekitar ketersediaan target, paling sering disebut sebagai sembilan (seperti ketersediaan 99,99%). Dengan tujuan ketersediaan, satu ukuran tidak cocok untuk semua. Sasaran ketersediaan perlu ditetapkan pada tingkat beban kerja versus menerapkan satu tujuan di semua beban kerja, memisahkan komponen non-kritis dari yang kritis.

Untuk kelangsungan operasi, point-in-time pengukuran berikut biasanya digunakan:

- Recovery Time Objective (RTO) — RTO adalah penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan. Nilai ini menentukan durasi yang dapat diterima di mana layanan terganggu.
- Recovery Point Objective (RPO) — RPO adalah jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terbaru dan gangguan layanan.

Mirip dengan menetapkan tujuan ketersediaan, RTO dan RPO juga harus didefinisikan pada tingkat beban kerja. Untuk mencapai kontinuitas operasi yang lebih agresif atau persyaratan ketersediaan yang tinggi, diperlukan peningkatan investasi. Meskipun demikian, tidak setiap aplikasi dapat menuntut atau membutuhkan tingkat ketahanan yang sama. Membuat mekanisme tiering dapat membantu membangun kerangka kerja untuk menyelaraskan pemilik bisnis dan TI dalam mengidentifikasi aplikasi yang paling menuntut berdasarkan dampak bisnis, dan meningkatkannya sesuai dengan itu. Contoh tiering dapat ditemukan di tabel berikut.

Tabel 1 — Contoh tingkatan ketahanan untuk SLA

Perjanjian Tingkat Layanan Ketersediaan (SLA)	Tingkat ketahanan	Downtime yang dapat diterima/tahun
99,99%	Platinum	52.60 menit

Perjanjian Tingkat Layanan Ketersediaan (SLA)	Tingkat ketahanan	Downtime yang dapat diterima/tahun
99,90%	Emas	8.77 jam
99,5%	Perak	1.83 hari

Tabel 2 — Contoh tingkatan ketahanan untuk RTO dan RPO

Tingkat	Max RTO	Max RPO	Kriteria	Biaya
Platinum	15 menit	lima menit	Beban kerja kritis misi	$$$
Emas	15 menit — enam jam	dua jam	Beban kerja penting, tetapi tidak penting untuk misi	$$
Perak	enam jam — beberapa hari	24 jam	Beban kerja yang tidak kritis	$

Saat merancang beban kerja untuk ketahanan, pemahaman tentang hubungan antara ketersediaan tinggi dan kontinuitas operasi diperlukan. Misalnya, jika beban kerja membutuhkan ketersediaan 99,99%, tidak lebih dari 53 menit downtime per tahun dapat ditoleransi. Diperlukan setidaknya lima menit untuk mendeteksi kegagalan dan sepuluh menit lagi bagi operator untuk terlibat, membuat keputusan tentang langkah-langkah pemulihan, dan melakukan langkah-langkah ini. Bukan hal yang aneh jika satu masalah membutuhkan waktu 30 hingga 45 menit untuk pulih. Dalam hal ini, memiliki strategi Multi-wilayah untuk memberikan contoh terisolasi yang menghilangkan dampak berkorelasi dapat memungkinkan operasi lanjutan dengan gagal dalam waktu terbatas, sambil memuji penurunan nilai awal secara independen. Di sinilah mendefinisikan RTO dan RPO yang sesuai diperlukan.

Untuk beban kerja kritis misi yang memiliki kebutuhan ketersediaan ekstrim (misalnya, ketersediaan 99,99% atau lebih tinggi) atau kontinuitas ketat persyaratan operasi yang hanya dapat dipenuhi dengan gagal masuk ke Wilayah lain, pendekatan Multi-wilayah mungkin tepat. Namun, persyaratan ini biasanya hanya berlaku untuk sebagian kecil dari portofolio beban kerja perusahaan yang memiliki

waktu pemulihan terbatas yang diukur dalam menit atau jam. Kecuali aplikasi memerlukan waktu pemulihan beberapa menit atau beberapa jam, menunggu gangguan Regional pada aplikasi untuk diperbaiki di Wilayah yang terkena dampak mungkin merupakan pendekatan yang lebih baik, dan biasanya selaras dengan beban kerja tingkat yang lebih rendah.

Sebelum menerapkan arsitektur Multi-region, pengambil keputusan bisnis dan tim teknis harus selaras dengan implikasi biaya, termasuk penggerak biaya operasional dan infrastruktur. Arsitektur Multi-wilayah yang khas dapat menimbulkan kenaikan biaya dua kali lipat dibandingkan pendekatan Single-region. Meskipun ada beberapa pola Multi-wilayah untuk kelangsungan bisnis, seperti berjalan dengan siaga panas, siaga hangat, dan lampu pilot, pola dengan risiko terendah untuk memenuhi tujuan pemulihan akan melibatkan menjalankan siaga panas, dan akan menggandakan biaya untuk beban kerja Anda.

Panduan utama

- Ketersediaan dan kelangsungan tujuan operasi seperti RTO dan RPO harus ditetapkan per beban kerja dan selaras dengan pemangku kepentingan bisnis dan TI.
- Sebagian besar ketersediaan dan kelangsungan tujuan operasi dapat dipenuhi dalam satu Wilayah. Untuk tujuan yang tidak dapat dipenuhi dengan satu Wilayah, Multi-wilayah harus dipertimbangkan, dengan pandangan yang jelas tentang pengorbanan antara biaya, kompleksitas, dan manfaat.

Multi-Region fundamental 2: Memahami data

Mengelola data adalah masalah non-sepele dengan arsitektur Multi-region. Jarak geografis antar Wilayah memberlakukan latensi yang tidak dapat dihindari, yang bermanifestasi sebagai waktu yang diperlukan untuk mereplikasi data di seluruh Wilayah. Pertukaran antara ketersediaan, konsistensi data, dan memperkenalkan urutan latensi yang lebih tinggi ke dalam beban kerja yang menggunakan arsitektur Multi-wilayah akan diperlukan. Baik menggunakan replikasi asinkron atau sinkron, Anda harus memodifikasi aplikasi untuk menangani perubahan perilaku yang diberlakukan oleh teknologi replikasi. Sangat sulit untuk mengambil aplikasi yang sudah ada yang dirancang untuk wilayah Tunggal dan menjadikannya Multi-wilayah karena tantangan seputar konsistensi dan latensi data. Memahami persyaratan konsistensi data dan pola akses data untuk beban kerja tertentu sangat penting untuk menimbang trade-off.

2a: Memahami persyaratan konsistensi data

Teorema CAP memberikan referensi untuk penalaran tentang pengorbanan antara konsistensi data, ketersediaan, dan partisi jaringan, yang hanya dua yang dapat dipenuhi pada saat yang sama untuk beban kerja. Multi-Region menurut definisi mencakup partisi jaringan antar Wilayah, jadi Anda harus memilih antara ketersediaan dan konsistensi.

Jika Anda memilih ketersediaan data di seluruh Wilayah, Anda tidak akan mengalami latensi yang signifikan selama penulisan transaksional, karena ada ketergantungan pada replikasi asinkron dari data yang dikomit antar Wilayah, sehingga konsistensi berkurang di seluruh Wilayah hingga replikasi selesai. Dengan replikasi asinkron, ketika ada kegagalan di Wilayah primer, ada kemungkinan besar penulisan menunggu replikasi dari Wilayah primer. Hal ini menyebabkan skenario di mana data terbaru tidak tersedia sampai replikasi dilanjutkan, dan proses rekonsiliasi diperlukan untuk menangani transaksi dalam penerbangan yang tidak mereplikasi dari Wilayah yang mengalami pemadaman.

Untuk beban kerja yang disukai replikasi asinkron, Anda dapat menggunakan layanan seperti Amazon Aurora dan Amazon DynamoDB, yang menyediakan replikasi lintas wilayah asinkron. Baik Amazon Aurora Global Database dan tabel global Amazon DynamoDB memiliki metrik CloudWatchAmazon default untuk membantu memantau kelambatan replikasi.

Rekayasa beban kerja untuk memanfaatkan arsitektur berbasis peristiwa merupakan manfaat bagi strategi Multi-wilayah karena itu berarti beban kerja dapat merangkul replikasi data asinkron, dan memungkinkan rekonstruksi status dengan memutar ulang peristiwa. Karena layanan streaming dan

pesan menyangga data payload pesan dalam satu Wilayah, proses failover/failback Regional harus menyertakan mekanisme untuk mengarahkan arus data masukan klien, serta merekonsiliasi muatan dalam penerbangan dan/atau tidak terkirim yang disimpan di Wilayah yang mengalami pemadaman.

Jika konsistensi dipilih, Anda akan mengalami latensi yang signifikan karena data direplikasi secara sinkron selama penulisan transaksional. Saat menulis ke beberapa Wilayah secara serempak, jika penulisan tidak berhasil di semua Wilayah, ketersediaan berpotensi diturunkan karena transaksi tidak akan dilakukan, dan perlu dicoba ulang. Percobaan ulang yang mencoba menulis data ke semua Wilayah secara serempak dilakukan dengan biaya latensi dengan setiap upaya. Pada titik tertentu, ketika percobaan ulang telah habis, keputusan perlu dibuat untuk gagal transaksi sepenuhnya, sehingga mengurangi ketersediaan, atau melakukan transaksi ke Wilayah yang tersedia saja, sehingga menyebabkan ketidakkonsistenan. Ada teknologi pembentuk kuorum seperti Paxos, yang dapat membantu mereplikasi dan melakukan data secara serempak, tetapi membutuhkan investasi pengembang yang signifikan.

Ketika penulisan melibatkan replikasi sinkron di beberapa Wilayah untuk memenuhi persyaratan konsistensi yang kuat, latensi tulis meningkat dengan urutan besarnya. Latensi tulis yang lebih tinggi bukanlah sesuatu yang biasanya dapat dipasang kembali ke dalam aplikasi tanpa perubahan signifikan. Idealnya, itu harus dipertimbangkan ketika aplikasi pertama kali dirancang. Untuk beban kerja Multi-wilayah di mana replikasi sinkron menjadi prioritas, solusi AWSMitra dapat membantu.

2b: Memahami pola akses data

Pola akses data beban kerja termasuk dalam salah satu jenis berikut: read-intensive atau write-intensive. Memahami karakteristik ini untuk beban kerja tertentu akan memandu pemilihan arsitektur Multi-wilayah yang sesuai.

Untuk beban kerja intensif baca seperti konten statis yang sepenuhnya hanya-baca, arsitektur Multi-wilayah aktif/aktif dapat dicapai tanpa kompleksitas yang signifikan. Menyajikan konten statis di tepi menggunakan Jaringan Distribusi Konten (CDN) memastikan ketersediaan dengan menyimpan konten yang paling dekat dengan pengguna akhir; menggunakan set fitur seperti failover Origin di Amazon CloudFront dapat membantu mencapai hal ini. Pilihan lainnya adalah menerapkan stateless compute di beberapa Wilayah dan menggunakan DNS untuk merutekan pengguna ke Wilayah terdekat untuk membaca konten. Rute 53 dengan kebijakan perutean geolokasi dapat digunakan untuk mencapai hal ini.

Untuk beban kerja intensif baca yang memiliki persentase pembacaan lebih besar daripada menulis, strategi global baca lokal, tulis dapat digunakan. Ini memerlukan semua penulisan pergi ke database di Wilayah tertentu dengan replikasi data asinkron ke semua Wilayah lain, dan pembacaan dapat

dilakukan di Wilayah mana pun untuk mencapai hal ini. Pendekatan ini membutuhkan beban kerja untuk merangkul konsistensi akhirnya, karena pembacaan lokal mungkin basi karena peningkatan latensi untuk replikasi penulisan lintas wilayah.

Aurora Global Database dapat membantu penyediaan Read Replicas di Wilayah siaga yang hanya dapat menangani semua lalu lintas baca secara lokal, dan satu datastore utama di Wilayah tertentu untuk menangani penulisan. Data direplikasi secara asinkron dari database primer ke basis data siaga (Baca Replika) dan database siaga dapat dipromosikan ke primer jika Anda perlu melakukan failover operasi ke Wilayah siaga. Jika beban kerja lebih cocok untuk model data non-relasional, DynamoDB dapat digunakan dalam pendekatan ini juga. Sekali lagi, beban kerja perlu merangkul konsistensi akhirnya, yang mungkin mengharuskannya ditulis ulang jika tidak dirancang untuk ini sejak awal.

Untuk beban kerja intensif penulisan, Wilayah utama harus dipilih dan kemampuan untuk melakukan failover ke Wilayah siaga harus direkayasa ke dalam beban kerja. Dibandingkan dengan pendekatan aktif/aktif, pendekatan primer/siaga tidak terlalu rumit. Ini karena untuk arsitektur aktif/aktif, beban kerja perlu ditulis ulang untuk menangani perutean cerdas ke Wilayah, membangun afinitas sesi, memastikan transaksi idempoten, dan menangani potensi konflik.

Sebagian besar beban kerja yang mencari ketahanan Multi-wilayah tidak memerlukan pendekatan aktif/aktif. Strategi sharding dapat digunakan untuk memberikan peningkatan ketahanan dengan membatasi radius ledakan gangguan di seluruh basis klien. Jika Anda dapat secara efektif menghancurkan basis klien, Wilayah primer yang berbeda dapat dipilih untuk setiap pecahan. Misalnya, jika Anda dapat membelah klien sehingga setengah dari klien disejajarkan dengan Wilayah Satu dan setengahnya disejajarkan dengan Wilayah Dua, memperlakukan Wilayah sebagai sel, pendekatan sel Multi-wilayah dapat dibuat, yang menghasilkan pengurangan radius ledakan dampak untuk beban kerja Anda.

Pendekatan sharding dapat dikombinasikan dengan pendekatan primer/siaga untuk memberikan kemampuan failover untuk pecahan. Proses failover yang diuji perlu direkayasa ke dalam beban kerja dan proses untuk rekonsiliasi data perlu direkayasa juga untuk memastikan konsistensi transaksional penyimpanan data setelah failover. Ini dibahas secara lebih rinci nanti dalam paper ini.

Panduan utama

- Ada kemungkinan besar bahwa penulisan yang tertunda untuk replikasi tidak akan dilakukan ke Wilayah siaga ketika ada kegagalan. Data tidak akan tersedia sampai replikasi dilanjutkan (dengan asumsi replikasi asinkron).

- Sebagai bagian dari failover, proses rekonsiliasi data akan diperlukan untuk memastikan status yang konsisten secara transaksional dipertahankan untuk datastores menggunakan replikasi asinkron.
- Ketika konsistensi yang kuat diperlukan, beban kerja perlu dimodifikasi untuk mentolerir latensi yang diperlukan dari datastore yang bereplikasi secara sinkron.

Multi-Region fundamental 3: Memahami dependensi beban kerja Anda

Beban kerja tertentu mungkin memiliki beberapa dependensi di Wilayah, seperti AWS layanan yang digunakan, dependensi internal, dependensi pihak ketiga, dependensi jaringan, sertifikat, kunci, rahasia, dan parameter. Untuk memastikan pengoperasian beban kerja selama skenario kegagalan, seharusnya tidak ada dependensi antara Wilayah primer dan Wilayah siaga; masing-masing harus dapat beroperasi secara independen satu sama lain. Untuk mencapai hal ini, semua dependensi dalam beban kerja harus diteliti untuk memastikan mereka tersedia di setiap Wilayah. Hal ini diperlukan karena kegagalan di Wilayah primer seharusnya tidak berdampak pada Wilayah siaga. Selain itu, pengetahuan tentang bagaimana beban kerja beroperasi ketika ketergantungan berada dalam keadaan terdegradasi atau sama sekali tidak tersedia sangat penting, sehingga solusi dapat direkayasa untuk menangani hal ini dengan tepat.

3a: layanan AWS

Saat merancang arsitektur Multi-wilayah, pemahaman tentang AWS layanan spesifik yang akan digunakan diperlukan. Aspek pertama adalah memahami fitur apa yang dimiliki layanan untuk mengaktifkan Multi-wilayah, dan apakah solusi harus direkayasa untuk mencapai tujuan Multi-wilayah. Misalnya, dengan Amazon Aurora dan Amazon DynamoDB, ada fitur untuk mereplikasi data secara asinkron ke Wilayah siaga. Dependensi AWS layanan apa pun harus tersedia di semua Wilayah tempat beban kerja akan dijalankan. Untuk memastikan layanan yang akan digunakan tersedia di Wilayah yang diinginkan, tinjau Daftar Layanan Wilayah AWS al.

3b: Dependensi internal dan pihak ketiga

Untuk setiap dependensi internal yang dimiliki beban kerja, pastikan itu tersedia dari Wilayah tempat beban kerja akan beroperasi. Misalnya, jika beban kerja terdiri dari banyak layanan mikro, berpengetahuan tentang semua layanan mikro yang terdiri dari kemampuan bisnis. Dari sana, pastikan bahwa semua layanan mikro tersebut dikerahkan di setiap Wilayah tempat beban kerja akan beroperasi.

Panggilan Lintas Wilayah antara layanan mikro dalam beban kerja tidak disarankan, dan isolasi Regional harus dipertahankan. Ini karena membuat dependensi lintas wilayah menambah risiko kegagalan yang berkorelasi, yang meniadakan manfaat yang Anda coba capai dengan implementasi

beban kerja Regional yang terisolasi. Dependensi lokal mungkin menjadi bagian dari beban kerja juga, jadi memahami bagaimana karakteristik integrasi ini dapat berubah jika Wilayah utama ingin berubah sangat penting. Misalnya, jika Wilayah siaga terletak lebih jauh dari lingkungan lokal, peningkatan latensi akan berdampak negatif.

Memahami solusi Perangkat Lunak sebagai Layanan (SaaS), kit pengembangan perangkat lunak (SDK), dan dependensi produk pihak ketiga lainnya, dan dapat menjalankan skenario di mana dependensi ini terdegradasi atau tidak tersedia akan memberikan lebih banyak wawasan tentang bagaimana rantai sistem beroperasi dan berperilaku di bawah mode kegagalan yang berbeda. Dependensi ini bisa berada dalam kode aplikasi dari bagaimana rahasia dikelola secara eksternal menggunakan AWS Secrets Manager, atau solusi vault pihak ketiga (seperti Hashicorp), hingga sistem otentikasi yang memiliki ketergantungan pada IAM Identity Center untuk login federasi.

Memiliki redundansi dalam hal dependensi dapat membantu meningkatkan ketahanan. Ada juga kemungkinan bahwa solusi SaaS atau ketergantungan pihak ketiga menggunakan primer yang sama dengan beban kerjaWilayah AWS. Jika ini masalahnya, Anda harus bekerja dengan vendor untuk menentukan apakah postur ketahanan mereka sesuai dengan persyaratan untuk beban kerja.

Selain itu, waspadai nasib bersama antara beban kerja dan dependensinya, seperti aplikasi pihak ketiga. Jika dependensi tidak tersedia di (atau dari) Wilayah sekunder setelah failover, beban kerja mungkin tidak pulih sepenuhnya.

3c: Mekanisme failover

Domain Name System (DNS) biasanya digunakan sebagai mekanisme failover untuk mengalihkan lalu lintas dari Region utama ke Region standby. Tinjau dan teliti secara kritis semua dependensi yang dibutuhkan mekanisme failover. Misalnya, jika beban kerja Anda menggunakan Amazon Route 53, memahami bahwa bidang kontrol di-host di US-East-1 berarti Anda mengambil dependensi pada bidang kontrol di Wilayah tertentu. Ini tidak direkomendasikan sebagai bagian dari mekanisme failover jika Wilayah utama adalah AS-Timur-1 juga. Jika mekanisme failover lain digunakan, pemahaman mendalam tentang skenario apa pun di mana ia tidak akan beroperasi seperti yang diharapkan diperlukan. Setelah pemahaman ini terbentuk, rencanakan kontingensi atau kembangkan mekanisme baru jika diperlukan. Tinjau Membuat Mekanisme Pemulihan Bencana Menggunakan Amazon Route 53 untuk mempelajari pendekatan yang dapat Anda gunakan untuk failover dengan sukses.

Sebagaimana dibahas di bagian ketergantungan internal, semua layanan mikro yang merupakan bagian dari kemampuan bisnis harus tersedia di setiap Wilayah di mana beban kerja digunakan.

Sebagai bagian dari strategi failover, kemampuan bisnis perlu gagal bersama untuk menghilangkan kemungkinan panggilan lintas wilayah. Atau, jika layanan mikro gagal secara independen, ini memperkenalkan potensi perilaku yang tidak diinginkan di mana layanan mikro berpotensi melakukan panggilan lintas wilayah, yang memperkenalkan latensi dan dapat menyebabkan beban kerja tidak tersedia jika terjadi batas waktu klien.

3d: Ketergantungan konfigurasi

Sertifikat, kunci, rahasia, dan parameter adalah bagian dari analisis ketergantungan yang diperlukan saat merancang untuk Multi-region. Jika memungkinkan, yang terbaik adalah melokalkan komponen-komponen ini di setiap Wilayah sehingga mereka tidak memiliki nasib bersama antar Wilayah untuk dependensi ini. Untuk sertifikat, kedaluwarsa harus bervariasi di antara mereka, dan jika mungkin, di setiap Wilayah, untuk menghindari skenario ketika sertifikat kedaluwarsa (dengan alarm diatur untuk memberitahukan sebelumnya) berdampak pada beberapa Wilayah.

Kunci enkripsi dan rahasia harus spesifik Wilayah juga. Dengan begitu, jika ada kesalahan dalam rotasi kunci atau rahasia, dampaknya terbatas pada Wilayah tertentu.

Terakhir, parameter beban kerja apa pun harus disimpan secara lokal agar beban kerja dapat diambil di Wilayah tertentu.

Panduan utama

- Arsitektur multi-wilayah mendapat manfaat dari pemisahan fisik dan logis antar Wilayah. Memperkenalkan dependensi lintas wilayah pada lapisan aplikasi merusak manfaat ini. Hindari dependensi seperti itu.
- Kontrol failover harus berfungsi tanpa dependensi pada Wilayah utama.
- Mengkoordinasikan failover pada kemampuan bisnis perlu dilakukan untuk menghilangkan kemungkinan peningkatan latensi dan ketergantungan panggilan lintas wilayah.

Multi-Region fundamental 4: Kesiapan operasional

Mengoperasikan beban kerja Multi-wilayah adalah tugas kompleks yang disertai dengan tantangan operasional yang spesifik untuk Multi-wilayah. Ini termasuk Akun AWS manajemen, proses penerapan retooled, membuat strategi observabilitas Multi-wilayah, membuat dan menguji failover dan runbook failback, dan kemudian mengelola biaya. Tinjauan Kesiapan Operasional (ORR) dapat membantu tim menyiapkan beban kerja untuk produksi, baik berjalan di satu Wilayah atau di beberapa Wilayah.

4a: manajemen Akun AWS

Untuk menerapkan beban kerja di seluruh wilayahWilayah AWS, pastikan bahwa semua kuota AWS layanan dalam akun berada dalam paritas di seluruh Wilayah. Pertama, pelajari semua AWS layanan yang merupakan bagian dari arsitektur, lihat penggunaan yang direncanakan di Wilayah siaga, lalu bandingkan dengan penggunaan saat ini. Dalam beberapa kasus, jika Wilayah siaga belum pernah digunakan sebelumnya, Anda dapat mereferensikan kuota layanan default untuk memahami titik awal. Kemudian, di semua layanan yang akan digunakan, minta peningkatan kuota menggunakan konsol Service Quotas (diperlukan login) atau API.

AWSPeran Identity and Access Management (IAM) perlu dikonfigurasi di setiap Wilayah untuk memastikan operator, peralatan otomatisasi, dan AWS layanan memiliki izin yang sesuai untuk sumber daya dalam Wilayah siaga. Peran isolasi regional mencapai isolasi Regional yang kami kejar untuk arsitektur Multi-wilayah. Pastikan izin ini ada sebelum ditayangkan dengan Wilayah siaga.

4b: Praktik penyebaran

Dengan kemampuan Multi-region, penyebaran beban kerja ke beberapa Wilayah bisa menjadi kompleks. AWS CloudFormationmembantu menyebarkan infrastruktur ke satu atau beberapa Wilayah, dan dapat disesuaikan sesuai dengan kebutuhan Anda. AWS CodePipelinembantu menyediakan pipeline integrasi/pengiriman berkelanjutan (CI/CD) yang hampir berkelanjutan, yang memiliki tindakan Lintas wilayah yang memungkinkan penyebaran ke Wilayah yang berbeda dari Wilayah tempat pipa berada. Ini, dikombinasikan dengan strategi penerapan yang kuat seperti biru/hijau, memungkinkan penerapan downtime minimum hingga nol.

Namun, penerapan kemampuan stateful bisa lebih kompleks ketika status aplikasi atau data tidak dieksternalisasi ke penyimpanan persisten. Dalam situasi ini, sesuaikan proses penyebaran dengan hati-hati agar sesuai dengan kebutuhan Anda. Rancang pipeline penyebaran dan proses untuk

diterapkan di satu Wilayah pada satu waktu, versus beberapa Wilayah secara bersamaan. Ini mengurangi kemungkinan kegagalan yang berkorelasi antar Daerah. Untuk mempelajari teknik yang digunakan Amazon untuk mengotomatiskan penerapan perangkat lunak, baca artikel Builder Library Mengotomatiskan penerapan yang aman dan hands-off.

4c: Observabilitas

Saat merancang untuk Multi-region, pertimbangkan bagaimana kesehatan semua komponen di setiap Wilayah akan dipantau untuk mendapatkan pandangan holistik tentang kesehatan Regional. Ini dapat mencakup metrik pemantauan untuk kelambatan replikasi, yang bukan merupakan pertimbangan untuk satu beban kerja Wilayah.

Saat membangun arsitektur Multi-wilayah, pertimbangkan untuk mengamati kinerja beban kerja dari Wilayah siaga juga. Ini termasuk pemeriksaan kesehatan dan kenari (pengujian sintetis) yang berjalan dari Wilayah siaga, memberikan pandangan luar tentang kesehatan primer. Selain itu, Anda dapat menggunakan Amazon CloudWatch Internet Monitor untuk memahami keadaan jaringan eksternal dan kinerja beban kerja Anda dari perspektif pengguna akhir. Demikian pula, Wilayah primer harus memiliki observabilitas yang sama untuk memantau Wilayah siaga. Burung kenari ini harus memantau metrik pengalaman pelanggan untuk mendapatkan kesehatan beban kerja secara keseluruhan. Hal ini diperlukan karena jika ada masalah di Wilayah primer, observabilitas di primer dapat terganggu dan akan berdampak pada kemampuan untuk menilai kesehatan beban kerja.

Dalam hal ini, mengamati di luar Wilayah itu dapat memberikan wawasan. Metrik ini harus digulung ke dasbor yang tersedia di setiap Wilayah, dan alarm dibuat di setiap Wilayah. Karena Amazon CloudWatch adalah layanan Regional, memiliki ini di kedua Wilayah adalah persyaratan. Data pemantauan ini akan digunakan untuk melakukan panggilan ke failover dari Primer ke Region standby.

4d: Proses, prosedur, dan pengujian

Waktu terbaik untuk menjawab pertanyaan, 'Kapan saya harus gagal? ' Jauh sebelum Anda membutuhkannya. Rencana kelangsungan bisnis termasuk orang, proses, dan teknologi semuanya harus didefinisikan dengan baik sebelum suatu masalah, dan diuji secara teratur. Tentukan kerangka keputusan pemulihan. Jika ada proses pemulihan yang dipraktikkan dengan baik dan waktu untuk pemulihan dipahami dengan baik, titik waktu untuk memulai proses pemulihan yang memenuhi target RTO melalui failover dapat dipilih. Titik waktu ini bisa segera setelah masalah dengan aplikasi di Wilayah utama diidentifikasi, atau bisa lebih jauh ke dalam peristiwa di mana opsi pemulihan dalam aplikasi di Wilayah telah habis, dan sekarang harus memulai failover untuk memenuhi RTO.

Sementara tindakan failover itu sendiri harus 100% otomatis, keputusan untuk mengaktifkan failover harus dibuat oleh manusia (biasanya sejumlah kecil individu yang telah ditentukan sebelumnya dalam organisasi). Juga, kriteria untuk memutuskan failover perlu didefinisikan dengan jelas dan dipahami secara global dengan organisasi. Proses ini dapat didefinisikan dan diselesaikan menggunakan runbook Manajer AWS Sistem, yang memungkinkan end-to-end otomatisasi lengkap dan memastikan konsistensi proses yang berjalan selama pengujian dan failover.

Runbook ini harus tersedia di Wilayah primer dan siaga untuk memulai proses failover atau failback. Setelah otomatisasi ini dilakukan, irama pengujian reguler harus ditentukan dan diikuti. Ini memastikan bahwa ketika ada peristiwa aktual, respons dijalankan pada proses yang terdefinisi dengan baik dan dipraktikkan yang dipercaya oleh organisasi. Penting juga untuk diingat toleransi yang ditetapkan untuk proses rekonsiliasi data. Konfirmasikan bahwa persyaratan RPO/RTO yang ditetapkan dipenuhi dengan proses yang diusulkan.

4e: Biaya dan kompleksitas

Implikasi biaya dari arsitektur Multi-region didorong oleh penggunaan infrastruktur yang lebih tinggi, overhead operasional, dan waktu sumber daya. Seperti disebutkan sebelumnya, biaya infrastruktur di Wilayah siaga mirip dengan biaya infrastruktur di Wilayah Primer ketika pra-penyediaan, menghasilkan dua kali biaya. Kapasitas penyediaan sehingga cukup untuk operasi sehari-hari, tetapi masih memiliki kapasitas penyangga yang cukup untuk mentolerir lonjakan permintaan — dan mengkonfigurasi batas yang sama di setiap Wilayah.

Selain itu, perubahan tingkat aplikasi mungkin diperlukan untuk berhasil berjalan dalam arsitektur Multi-wilayah jika Anda mengadopsi arsitektur aktif-aktif, yang dapat memakan waktu dan sumber daya intensif untuk merancang dan mengoperasikan. Minimal, organisasi perlu meluangkan waktu untuk memahami ketergantungan teknis dan bisnis di setiap Wilayah, dan merancang proses failover dan failback.

Tim juga harus melalui latihan failover dan failback normal untuk merasa nyaman dengan runbook yang akan digunakan selama acara. Meskipun sangat penting dan penting untuk mendapatkan hasil yang diharapkan dari investasi Multi-wilayah, latihan ini mewakili biaya peluang, dan mengambil waktu dan sumber daya dari kegiatan lain.

Panduan utama

- AWSkuota layanan perlu ditinjau dan paritas di semua Wilayah di mana beban kerja akan beroperasi.

- Proses penyebaran harus menargetkan satu Wilayah pada satu waktu, bukan beberapa Wilayah secara bersamaan.
- Metrik tambahan seperti lag replikasi perlu dipantau, dan khusus untuk skenario Multi-wilayah.
- Memperluas pemantauan untuk beban kerja di luar Wilayah utama. Metrik pengalaman pelanggan harus dipantau per Wilayah, dan diukur dari luar setiap Wilayah tempat beban kerja berjalan.
- Failover dan failback perlu diuji secara teratur. Pastikan implementasi runbook tunggal untuk proses failover dan failback yang digunakan selama pengujian dan acara langsung. Runbook untuk pengujian dan acara langsung tidak bisa berbeda.

Kesimpulan

Buku putih ini membahas kasus penggunaan umum untuk Multi-wilayah, dasar-dasar tentang cara menerapkan arsitektur Multi-wilayah, dan implikasi dari pendekatan ini. Dasar-dasar ini dapat diterapkan pada beban kerja apa pun dan digunakan sebagai kerangka kerja untuk membantu dalam pengambilan keputusan tentang apakah arsitektur Multi-wilayah adalah pendekatan yang tepat untuk bisnis tertentu atau tidak.

Kontributor

Kontributor dokumen ini meliputi:

Kontributor teknis:

- John Formento, Jr., Arsitek Solusi Utama, Tim Multi-Wilayah AWS

Kontributor editorial:

- Lisi Lewis, Sr. Manajer, Pemasaran Produk

Bacaan lebih lanjut

Untuk informasi tambahan, lihat:

- Pola Ketahanan Multi-AZ Tingkat Lanjut (whitepaper) AWS
- Pilar Keandalan - Kerangka AWS Well-Architected
- Ketersediaan dan Selanjutnya: Memahami dan Meningkatkan Ketahanan Sistem Terdistribusi di AWS (AWSwhitepaper)
- AWSBatas Isolasi Kesalahan (AWSwhitepaper)

Revisi dokumen

Untuk diberitahu tentang pembaruan pada whitepaper ini, berlangganan RSS feed.

Perubahan	Deskripsi	Tanggal
Dokumen diterbitkan	Publikasi pertama.	Desember 20, 2022

Pemberitahuan

Pelanggan bertanggung jawab untuk membuat penilaian independen mereka sendiri atas informasi dalam dokumen ini. Dokumen ini: (a) hanya untuk tujuan informasi, (b) mewakili penawaran dan praktik AWS produk saat ini, yang dapat berubah tanpa pemberitahuan, dan (c) tidak membuat komitmen atau jaminan apa pun dari AWS dan afiliasinya, pemasok, atau pemberi lisensinya. AWSproduk atau layanan disediakan “sebagaimana adanya” tanpa jaminan, representasi, atau kondisi apa pun, baik tersurat maupun tersirat. Tanggung jawab dan kewajiban AWS kepada pelanggannya dikendalikan oleh AWS perjanjian, dan dokumen ini bukan bagian dari, juga tidak mengubah, perjanjian apa pun antara AWS dan pelanggannya.

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.