Stabilitas statis menggunakan Availability Zone

Becky Weiss, Mike Furr

Di Amazon, layanan yang kami bangun harus memenuhi target ketersediaan yang benar-benar tinggi. Ini berarti kami harus berpikir dengan cermat tentang dependensi yang diambil oleh sistem kami. Kami merancang sistem kami agar tetap tangguh bahkan saat dependensi mengalami kerusakan. Dalam artikel ini, kami akan mendefinisikan pola yang kami gunakan, yang disebut *stabilitas statis* untuk mencapai tingkat ketangguhan ini. Kami akan menunjukkan kepada Anda bagaimana kami menerapkan konsep ini ke Availability Zone, sebuah blok bangunan infrastruktur utama di AWS, dan karena itu menjadi dependensi landasan untuk semua layanan yang kami bangun.

Dalam desain yang stabil secara statis, seluruh sistem terus bekerja bahkan saat dependensi mengalami gangguan. Mungkin sistem tidak melihat informasi terbaru (misalnya hal-hal baru, hal yang dihapus, atau hal yang dimodifikasi) yang dependensinya seharusnya telah dikirim. Meski demikian, semua yang dilakukannya sebelum dependensi mengalami gangguan akan terus bekerja terlepas dari dependensi yang mengalami kerusakan. Kami akan menjelaskan bagaimana kami membangun Amazon Elastic Compute Cloud (EC2) agar stabil secara statis. Kemudian kami akan memberikan dua arsitektur contoh yang stabil secara statis, yang menurut kami berguna untuk membangun sistem wilayah yang tersedia luas di atas Availability Zone.

Terakhir, kita akan membahas lebih dalam tentang beberapa filosofi desain di balik Amazon EC2, termasuk bagaimana Amazon EC2 dirancang untuk menyediakan dependensi Availability Zone pada tingkat perangkat lunak. Selain itu, kita akan membahas beberapa tradeoff yang menjadi bagian dari membangun layanan dengan pilihan arsitektur ini.

Peran Availability Zone

Availability Zone adalah bagian dari Wilayah AWS yang secara logis terisolasi: Setiap Wilayah AWS memiliki beberapa Availability Zone yang dirancang untuk beroperasi secara independen. Availability Zone dipisahkan secara fisik oleh jarak bermakna untuk melindungi dari dampak terkorelasi dari potensi masalah, misalnya sambaran petir, tornado, dan gempa bumi. Mereka tidak berbagi daya atau infrastruktur lainnya, tetapi terhubung dengan satu sama lain menggunakan jaringan serat optik pribadi yang cepat dan terenkripsi untuk memungkinkan aplikasi dengan cepat mengalami fail over tanpa gangguan. Dengan kata lain, Availability Zone menyediakan lapisan abstraksi terhadap isolasi infrastruktur kami. Layanan yang memerlukan Availability Zone memungkinkan pemanggil memberitahu AWS tempat untuk menyediakan infrastruktur secara fisik dalam Wilayah agar dapat mengambil keuntungan dari kemandirian ini. Di Amazon, kami telah mengembangkan layanan AWS wilayah yang memanfaatkan kemandirian zona ini untuk mencapai target ketersediaan tingginya sendiri. Layanan seperti Amazon DynamoDB, Amazon Simple Queue Service (SQS), dan Amazon Simple Storage Service (S3) merupakan contoh dari layanan wilayah tersebut.

Saat berinteraksi dengan layanan AWS yang menyediakan infrastruktur cloud di dalam Amazon Virtual Private Cloud (VPC), banyak dari layanan ini yang mengharuskan pemanggil untuk menetapkan tidak saja Wilayah, tetapi juga Availability Zone. Availability Zone sering ditetapkan secara implisit dalam argumen subnet yang diperlukan, misalnya saat meluncurkan instans EC2, menyediakan database Amazon Relational Database Service (RDS), atau membuat klaster Amazon ElastiCache. Meski umum memiliki beberapa subnet dalam Availability Zone, satu subnet sepenuhnya menempati satu Availability Zone, jadi dengan menyediakan argumen subnet, pemanggil juga secara implisit menyediakan Availability Zone untuk digunakan.

Stabilitas statis

Saat membangun sistem di atas Availability Zone, satu pelajaran yang kami pelajari adalah mempersiapkan diri menghadapi ketidaksesuaian sebelum terjadi. Sebuah pendekatan yang kurang efektif mungkin adalah dengan menerapkan beberapa Availability Zone dengan harapan bahwa, jika terjadi ketidaksesuaian dalam satu Availability Zone, layanan akan ditingkatkan skalanya (mungkin menggunakan AWS Auto Scaling) dalam Availability Zones lainnya dan dipulihkan ke kesehatan penuh. Pendekatan ini kurang efektif karena mengandalkan reaksi terhadap ketidaksesuaian saat terjadi, bukannya mempersiapkan diri untuk ketidaksesuaian tersebut sebelum terjadi. Dengan kata lain, pendekatan ini tidak memiliki stabilitas statis. Di sisi lain, sebuah layanan yang lebih efektif dan stabil secara statis akan menyediakan insfrastrukturnya secara berlebihan ke titik infrastruktur tersebut akan terus beroperasi dengan benar tanpa perlu meluncurkan instans EC2 baru, bahkan jika Availability Zone akan mengalami kerusakan.

Untuk mengilustrasikan properti stabilitas statis dengan lebih baik, mari kita lihat Amazon EC2, yang dirancang sesuai dengan prinsip tersebut.

Layanan Amazon EC2 terdiri dari bidang kontrol dan bidang data. “Bidang kontrol” dan “bidang data” merupakan istilah khusus dari jaringan, tetapi kami menggunakannya dalam semua hal dalam AWS. *Bidang kontrol* adalah mesin yang terlibat dalam pembuatan perubahan dalam sistem—menambahkan sumber daya, menghapus sumber daya, memodifikasi sumber daya—dan mengatur perubahan tersebut agar tersebar ke mana pun yang diperlukan agar memiliki dampak. Berkebalikan dengan *bidang data*, yang merupakan urusan sehari-hari dari sumber daya tersebut, yang dibutuhkannya untuk dapat berfungsi.

Di Amazon EC2, bidang kontrol adalah semua hal yang terjadi saat EC2 meluncurkan instans baru. Logika bidang kontrol menarik semua hal yang dibutuhkan untuk instans EC2 baru dengan menjalankan sejumlah tugas. Berikut adalah beberapa contohnya:

- Menemukan server fisik untuk komputasi serta tetap mempertimbangkan persyaratan grup penempatan dan tenancy VPC.
- Mengalokasikan antarmuka jaringan di luar subnet VPC.
- Mempersiapkan volume Amazon Elastic Block Store (EBS).
- Menghasikan kredensial peran AWS Identity and Access Management (IAM).
- Menginstal aturan Grup Keamanan.
- Menyimpan hasil dalam penyimpanan data berbagai layanan downstream.
- Menyebarkan konfigurasi yang dibutuhkan ke server di VPC dan edge jaringan sesuai kebutuhan.

Di sisi lain, bidang data Amazon EC2 memastikan instans EC2 tetap berjalan seperti seharusnya, melakukan tugas seperti berikut:

- Merutekan paket sesuai tabel rute VPC.
- Membaca dan menulis dari volume Amazon EBS.
- Dan sebagainya.

Seperti yang umum ditemukan pada bidang data dan bidang kontrol, bidang data Amazon EC2 jauh lebih sederhana daripada bidang kontrolnya. Sebagai hasil dari kesederhanaan relatifnya, desain bidang data Amazon EC2 menarget ketersediaan yang lebih tinggi daripada bidang kontrol Amazon EC2.

Yang juga penting, bidang data Amazon EC2 telah secara cermat dirancang agar stabil secara statis saat menghadapi peristiwa ketersediaan bidang kontrol (misalnya, ketidaksesuaian dalam kemampuan meluncurkan instans EC2). Sebagai contoh, untuk menghindari gangguan dalam konektivitas jaringan, bidang data Amazon EC2 dirancang agar mesin fisik tempat Amazon EC2 berjalan memiliki akses lokal ke semua informasi yang dibutuhkannya untuk merutekan paket ke titik di dalam dan di luar VPC-nya. Ketidaksesuaian bidang kontrol Amazon EC2 berarti bahwa selama peristiwa, server fisik mungkin tidak melihat pembaruan seperti instans EC2 baru ditambahkan ke VPC, atau aturan Grup Keamanan baru. Meski demikian, lalu lintas yang berhasil diterima dan dikirimnya sebelum peristiwa akan terus berfungsi.

Konsep bidang kontrol, bidang data, dan stabilitas statis berlaku sangat luas, bahkan di luar Amazon EC2. Kemampuan untuk menguraikan sistem ke dalam bidang kontrol dan bidang datanya dapat menjadi alat konseptual yang berguna untuk merancang layanan yang tersedia luas, karena sejumlah alasan:

- Ketersediaan bidang data umumnya jauh lebih penting bagi kesuksesan pelanggan layanan daripada bidang kontrol. Misalnya, ketersediaan berkelanjutan dan fungsi yang tepat dari instans EC2, setelah berjalan, semakin penting untuk sebagian besar pelanggan AWS daripada kemampuan untuk meluncurkan instans EC2 baru.
- Umumnya bidang data beroperasi pada volume yang lebih tinggi (seringkali berdasarkan magnitudonya) daripada bidang kontrolnya. Karena itu, sebaiknya membiarkan mereka tetap terpisah agar masing-masing dapat diskalakan sesuai dimensi penskalaan relevannya sendiri.
- Kami menemukan bahwa selama bertahun-tahun bidang kontrol sistem cenderung memiliki lebih banyak bagian bergerak daripada bidang datanya, sehingga secara statistik lebih cenderung mengalami kerusakan karena alasan tersebut.

Mempertimbangkan semua hal itu, praktik terbaik kami adalah memisahkan sistem di seluruh batasan bidang kontrol dan data.

Untuk mencapai pemisahan ini pada praktiknya, kami menerapkan prinsip stabilitas statis. Bidang data umumnya bergantung pada data yang muncul dari bidang kontrol. Meski demikian, untuk mencapai target ketersediaan yang lebih tinggi, bidang data mempertahankan status yang sudah ada dan terus bekerja bahkan saat menghadapi ketidaksesuaian bidang kontrol. Bidang data mungkin tidak menerima pembaruan selama periode ketidaksesuaian, tetapi semua hal yang telah berfungsi sebelumnya akan terus berfungsi.

Sebelumnya kami telah menjelaskan bahwa skema yang memerlukan penggantian instans EC2 dalam merespons ketidaksesuaian Availability Zone adalah pendekatan yang kurang efektif. Bukan karena kami tidak dapat meluncurkan instans EC2 baru. Ini karena dalam merespons ketidaksesuaian, sistem harus mengambil dependensi langsung untuk jalur pemulihan pada bidang kontrol Amazon EC2, serta semua sistem spesifik aplikasi yang dibutuhkan untuk instans baru mulai menjalankan pekerjaan yang berguna. Bergantung pada aplikasinya, dependensinya ini dapat menyertakan sejumlah langkah, seperti mengunduh konfigurasi runtime, mendaftarkan instans dengan layanan penemuan, mendapatkan kredensial, dll. Sistem bidang kontrol pada dasarnya lebih rumit daripada sistem dalam bidang data, dan memiliki kemungkinan yang lebih besar tidak berperilaku dengan benar saat seluruh sistem mengalami ketidaksesuaian.

Pola stabilitas statis

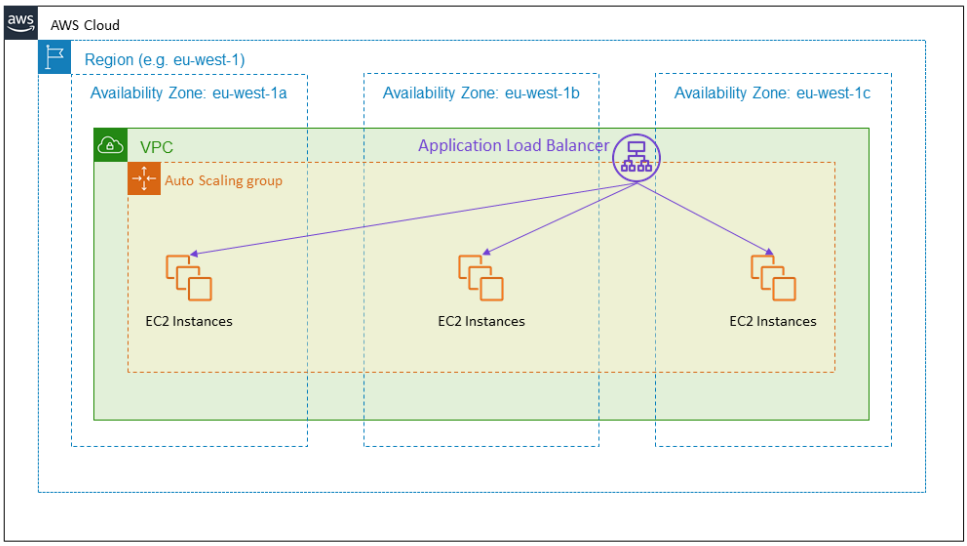
Dalam bagian ini, kami akan memperkenalkan dua pola tingkat tinggi yang kami gunakan di AWS untuk merancang sistem untuk ketersediaan tinggi dengan memanfaatkan stabilitas statis. Masing-

masing berlaku untuk kumpulan situasinya sendirinya, tetapi sama-sama memanfaatkan abstraksi Availability Zone.

Contoh aktif-aktif pada Availability Zone: Layanan dengan muatan seimbang

Beberapa layanan AWS secara internal terdiri dari armada instans EC2 tanpa status yang secara horizontal dapat diskalakan atau kontainer Amazon Elastic Container Service (ECS). Kami menjalankan layanan ini dalam Grup Auto Scaling di tiga atau lebih Availability Zone. Selain itu, kapasitas pengadaan berlebih layanan ini sehingga, bahkan jika seluruh Availability Zone mengalami ketidaksesuaian, server dalam sisa Availability Zone dapat membawa muatannya. Misalnya, jika kami menggunakan tiga Availability Zone, kami menyediakan secara berlebih sebesar 50 persen. Dengan kata lain, kami menyediakan berlebih sehingga setiap Availability Zone beroperasi hanya pada 66 persen tingkat yang telah kami uji dengan muatan.

Contoh yang paling umum adalah layanan HTTPS bermuatan seimbang. Diagram berikut menampilkan Application Load Balancer yang menghadap publik, yang menyediakan layanan HTTPS. Target penyeimbang muatan adalah grup Auto Scaling yang mencakup tiga Availability Zone di Wilayah ue-barat-1. Ini adalah contoh dari ketersediaan tinggi aktif-aktif menggunakan Availability Zone.



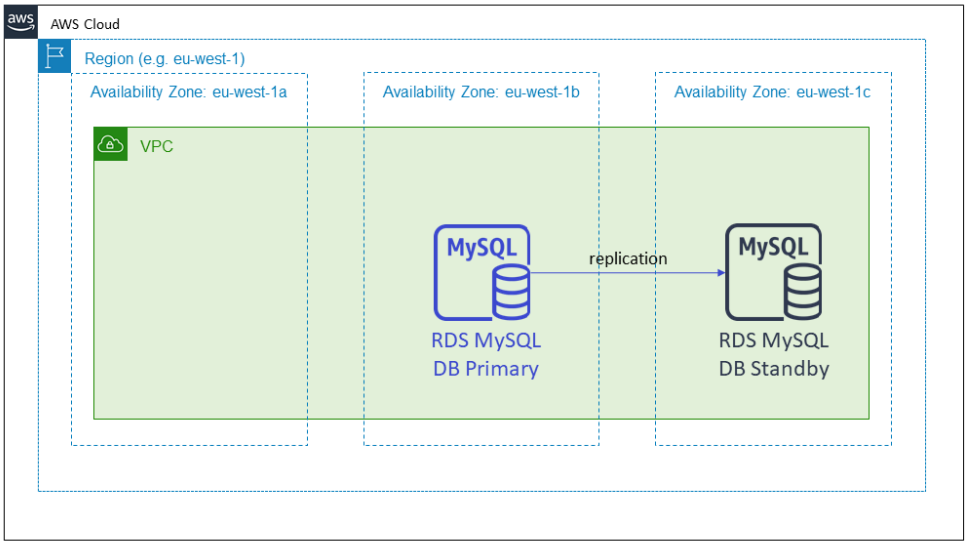
Jika terjadi ketidaksesuaian Availability Zone, arsitektur yang ditampilkan dalam diagram sebelumnya tidak memerlukan tindakan apa pun. Instans EC2 dalam Availability Zone yang mengalami ketidaksesuaian akan mulai gagal dalam pemeriksaan kesehatan, dan Application Load Balancer akan mengalihkan lalu lintas dari instans tersebut. Faktanya, layanan Elastic Load Balancing dirancang sesuai prinsip ini. Layanan ini telah menyediakan kapasitas penyeimbangan muatan yang cukup untuk menahan ketidaksesuaian Availability Zone tanpa perlu meningkatkan skala.

Kami juga menggunakan pola ini bahkan jika tidak terdapat penyeimbang muatan atau layanan HTTPS. Misalnya, armada instans EC2 yang memproses pesan dari antrean Amazon Simple Queue Service (SQS) dapat mengikuti pola ini juga. Instans diterapkan dalam grup Auto Scaling di beberapa Availability Zone, secara tepat disediakan berlebih. Jika Availability Zone mengalami kerusakan, layanan tidak akan melakukan apa pun. Instans yang mengalami kerusakan berhenti melakukan pekerjaannya, dan instans lainnya akan melanjutkan pekerjaan tersebut.

Contoh aktif-siaga pada Availability Zone: Database relasional

Beberapa layanan yang kami buat adalah layanan stateful dan memerlukan satu node utama atau pemimpin untuk mengoordinasikan pekerjaan. Contoh dari ini adalah layanan yang menggunakan database relasional, seperti Amazon RDS dengan mesin database MySQL atau Postgres. Penyiapan ketersediaan tinggi umum untuk jenis database relasional ini memiliki instans utama, yaitu instans yang menjadi tujuan semua penulisan, dan kandidat siaga. Kita juga mungkin memiliki replika baca tambahan, yang tidak ditampilkan dalam diagram berikut. Saat kita bekerja dengan infrastruktur stateful seperti ini, akan ada node siaga hangat dalam Availability Zone yang berbeda dari Availability Zone node utama.

Diagram berikut menampilkan database Amazon RDS. Jika kita menyediakan database dengan Amazon RDS, ini akan memerlukan grup subnet. *Grup subnet* adalah sebuah set subnet yang mencakup beberapa Availability Zone ke dalam instans database yang akan disediakan. Amazon RDS menempatkan kandidat siaga dalam beberapa Availability Zone dari node utama. Ini adalah contoh dari ketersediaan tinggi aktif-siaga menggunakan Availability Zone.



Seperti hal dengan contoh aktif-aktif tanpa status, saat Availability Zone dengan node utama mengalami ketidaksesuaian, layanan stateful tidak melakukan apa pun dengan infrastruktur. Untuk layanan yang menggunakan Amazon RDS, RDS akan mengelola failover dan mengarahkan ulang nama DNS ke utama baru dalam Availability Zone yang berfungsi. Pola ini juga berlaku untuk penyiapan aktif-siaga lain, bahkan jika mereka tidak menggunakan database relasional. Secara

khusus, kami menerapkan ini ke sistem dengan arsitektur klaster yang memiliki node pemimpin. Kami menerapkan klaster ini di seluruh Availability Zone dan memilih node pemimpin baru dari kandidat siaga, bukan meluncurkan pengganti “tepat waktu”.

Persamaan yang dimiliki kedua pola adalah keduanya telah menyediakan kapasitas yang mereka perlukuan saat terjadi ketidaksesuaian Availability Zone sebelum terjadi ketidaksesuaian aktual apa pun. Dalam kedua kasus ini adalah layanan mengambil dependensi bidang kontrol yang disengaja, seperti penyediaan infrastruktur baru atau pembuatan modifikasi, sebagai respons dari ketidaksesuaian Availability Zone.

Di balik layar: Stabilitas statis di dalam Amazon EC2

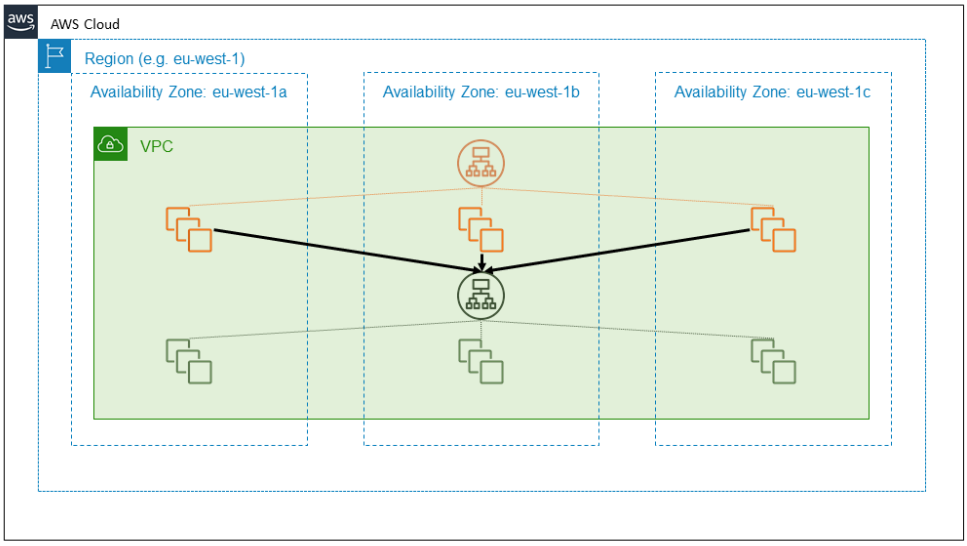
Bagian akhir artikel ini akan menyelam lebih dalam ke arsitektur Availability Zone yang tangguh, mencakup beberapa cara di mana kami mengikuti prinsip independensi Availability Zone di Amazon EC2. Memahami beberapa konsep ini sangat bermanfaat saat kami membangun layanan yang tidak hanya harus memiliki ketersediaan tinggi, tetapi juga harus menyediakan infrastruktur di mana layanan lain juga dapat memiliki ketersediaan tinggi. Amazon EC2, sebagai penyedia infrastruktur AWS tingkat rendah, merupakan infrastruktur yang dapat aplikasi gunakan agar memiliki ketersediaan yang tinggi. Ada saat ketika sistem lain juga ingin mengadopsi strategi tersebut.

Kami mengikuti prinsip independensi Availability Zone pada Amazon EC2 dalam praktik penerapan kami. Di Amazon EC2, perangkat lunak diterapkan ke server fisik yang me-host instans EC2, perangkat edge, resolver DNS, komponen bidang kontrol di jalur peluncuran instans EC2, dan banyak komponen lain yang diandalkan oleh instans EC2. Penerapan ini mengikuti kalender penerapan zona. Ini berarti bahwa dua Availability Zone di Wilayah yang sama akan menerima penerapan yang ditentukan pada hari yang berbeda. Di seluruh AWS, kami menggunakan peluncuran penerapan bertahap. Contohnya, kami mengikuti praktik terbaik (terlepas dari jenis layanan yang kami terapkan) dengan pertama-tama menerapkan one-box, lalu 1/N server, dll. Namun, dalam kasus layanan tertentu seperti pada Amazon EC2, penerapan kami dilakukan satu langkah lebih maju dan secara sengaja disejajarkan dengan batas Availability Zone. Dengan cara itu, masalah pada penerapan memengaruhi satu Availability Zone dan diluncurkan kembali serta diperbaiki. Ini tidak memengaruhi Availability Zone lain, sehingga terus berfungsi secara normal.

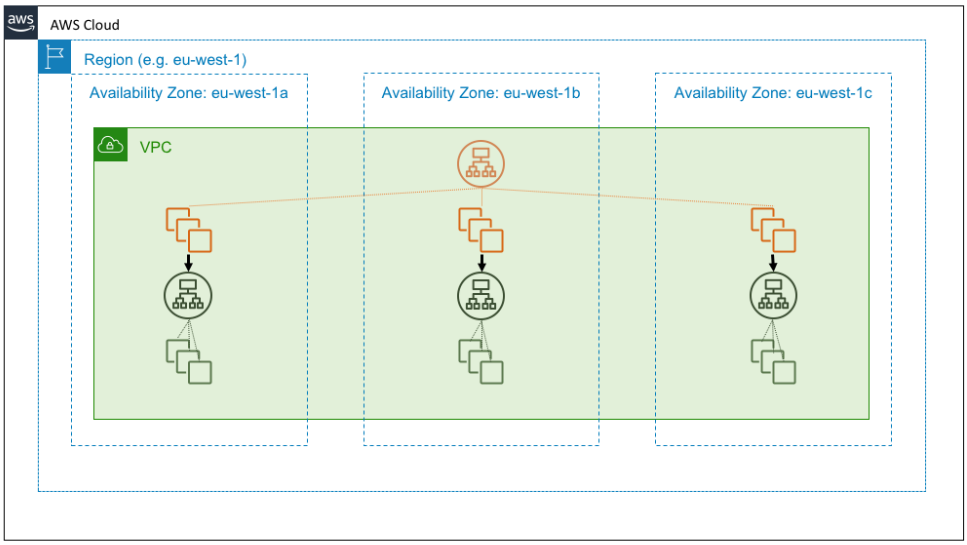
Cara lain kami dalam menggunakan prinsip independensi Availability Zone saat membangun di Amazon EC2 adalah dengan mendesain seluruh aliran paket agar tetap berada di dalam Availability Zone, tidak melintasi batas. Poin kedua ini—ketika lalu lintas jaringan dibuat tetap lokal ke Availability Zone—layak untuk dijelajahi secara lebih detail. Ini adalah ilustrasi yang menarik tentang bagaimana kami berpikir secara berbeda saat membangun sistem wilayah dengan ketersediaan tinggi yang merupakan konsumen Availability Zone independen (yaitu, menggunakan jaminan independensi Availability Zone sebagai dasar untuk membangun layanan dengan ketersediaan tinggi), berlawanan dengan saat kami menyediakan insfratruktur independen Availability Zone untuk konsumen lain yang akan memungkinkan mereka membangun untuk ketersediaan tinggi.

Diagram berikut mengilustrasikan layanan eksternal dengan ketersediaan tinggi, ditampilkan dalam warna oranye, yang mengandalkan layanan internal lain, ditampilkan dalam warna hijau. Desain yang lugas memperlakukan kedua layanan ini sebagai konsumen Availability Zone EC2 independen. Tiap layanan berwarna oranye dan hijau digawangi oleh Application Load Balancer,

dan tiap layanan memiliki armada host backend yang disediakan dengan baik dan tersebar di ketiga Availability Zone. Satu layanan wilayah dengan ketersediaan tinggi memanggil layanan wilayah lain yang juga memiliki ketersediaan tinggi. Ini adalah desain yang sederhana, dan bagi banyak layanan yang kami bangun, ini adalah desain yang baik.



Namun, anggaplah bahwa layanan hijau adalah layanan dasar. Hasilnya, anggap saja ini dimaksudkan untuk tidak hanya memiliki ketersediaan tinggi, tetapi juga berfungsi sebagai dasar untuk menyediakan independensi Availability Zone. Dalam kasus ini, kami mungkin merancangnya sebagai tiga instans layanan zona lokal dengan mengikuti praktik penerapan yang memerhatikan Availability Zone. Diagram berikut mengilustrasikan layanan di mana layanan wilayah dengan ketersediaan tinggi memanggil layanan zona dengan ketersediaan tinggi.

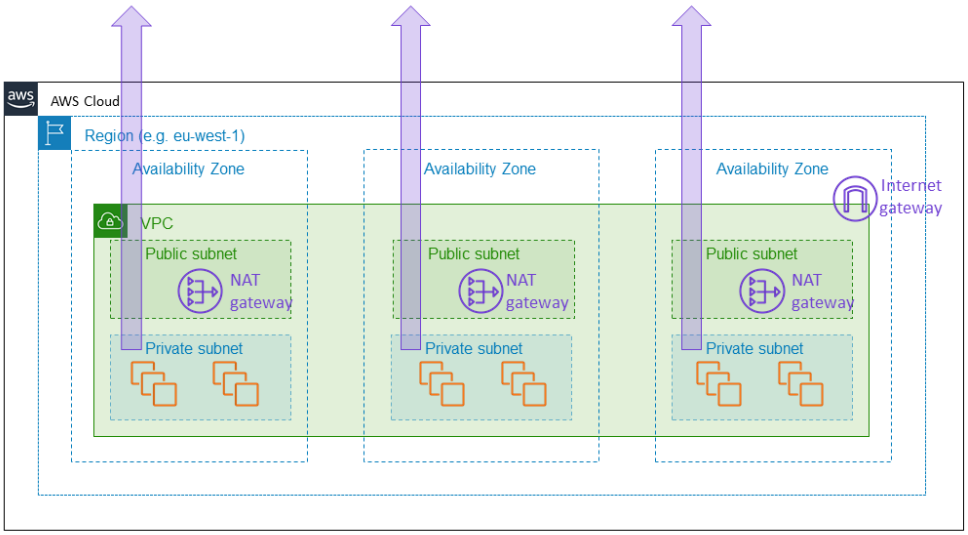


Alasan mengapa kami mendesain layanan dasar menjadi independen Availability Zone disimpulkan dengan aritmetika sederhana. Anggaplah Availability Zone tidak sesuai. Untuk kegagalan hitam dan putih, Application Load Balancer secara otomatis akan menjauh dari node yang terpengaruh. Namun, tidak semua kegagalan terlihat dengan jelas. Mungkin ada kegagalan abu-abu, seperti bug di perangkat lunak, yang tidak akan dapat dilihat penyeimbang muatan dalam pemeriksaan kesehatan dan menanganinya dengan baik.

Pada contoh sebelumnya, ketika satu layanan wilayah dengan ketersediaan tinggi memanggil layanan wilayah lain yang juga memiliki ketersediaan tinggi, jika permintaan dikirim melalui sistem, lalu dengan beberapa asumsi penyederhanaan, peluang permintaan menghindari Availability Zone yang tidak sesuai adalah 2/3 * 2/3 = 4/9. Artinya, permintaan tersebut menjadi lebih buruk daripada peluang menghindari kejadian. Sebaliknya, jika kami membangun layanan berwarna hijau untuk menjadi layanan zona seperti pada contoh saat ini, host di layanan berwarna oranye dapat memanggil titik akhir berwarna hijau di Availability Zone yang sama. Dengan arsitektur ini, peluang menghindari Availability Zone yang tidak sesuai adalah 2/3. Jika layanan N merupakan bagian dari jalur panggilan ini, maka angka tersebut digeneralisasikan ke (2/3)^N untuk layanan wilayah N versus konstan tersisa pada 2/3 untuk layanan zona N.

Oleh karena alasan ini kami membangun gateway NAT Amazon EC2 sebagai layanan zona. Gateway NAT adalah fitur Amazon EC2 yang mengizinkan lalu lintas internet keluar dari subnet pribadi dan muncul tidak sebagai gateway wilayah di seluruh VPC, tetapi sebagai sumber daya zona yang diinstankan pelanggan secara terpisah per Availability Zone seperti yang ditampilkan pada diagram berikut. Gateway NAT berada di jalur konektivitas internet untuk VPC, dan merupakan bagian bidang data instans EC2 mana pun di dalam VPC tersebut. Jika ada ketidaksesuaian konektivitas di satu Availability Zone, kami ingin menjaga agar ketidaksesuaian itu tetap berada di dalam Availability Zone, tidak menyebarkannya ke zona lain. Pada akhirnya, kami ingin pelanggan yang membangun arsitektur yang serupa dengan yang telah disebutkan di awal artikel ini (yaitu, dengan

menyediakan armada di ketiga Availability Zone dengan kapasitas yang cukup pada dua Availability Zone untuk membawa muatan penuh) untuk mengetahui bahwa Availability Zone lain akan sepenuhnya tidak terpengaruh oleh apa pun yang terjadi dalam Availability Zone yang tidak sesuai. Satu-satunya cara bagi kami untuk melakukan ini adalah dengan memastikan bahwa seluruh komponen dasar—seperti gateway NAT—benar-benar berada di dalam Availability Zone.



Pilihan ini memunculkan kerumitan tambahan. Bagi kami di Amazon EC2, kerumitan tambahan muncul dalam bentuk pengelolaan zona, bukan lingkungan layanan wilayah. Bagi pelanggan gateway NAT, kerumitan tambahan muncul dalam bentuk adanya beberapa gateway NAT dan tabel rute, untuk digunakan di Availability Zone yang berbeda pada VPC. Kerumitan tambahan ini cukup wajar karena gateway NAT itu sendiri merupakan layanan dasar, bagian dari bidang data Amazon EC2 yang seharusnya menyediakan jaminan ketersediaan zona.

Ada satu pertimbangan lagi yang kami buat saat membangun layanan yang independen pada Availability Zone, yaitu durabilitas data. Meski tiap arsitektur zona yang telah dijelaskan di awal menampilkan seluruh tumpukan yang berada di dalam satu Availability Zone, kami mereplikasi status keras apa pun di beberapa Availability Zone untuk tujuan pemulihan bencana. Sebagai contoh, umumnya kami menyimpan cadangan database berkala di Amazon S3 dan mempertahankan replika baca pada data kami yang disimpan di seluruh batas Availability Zone. Replika ini tidak perlu berfungsi bagi Availability Zone utama. Alih-alih, mereka memastikan bahwa kami menyimpan data penting pelanggan atau bisnis di beberapa lokasi.

Saat merancang arsitektur berbasis layanan yang akan dijalankan di AWS, kami telah belajar menggunakan satu atau dua pola ini, atau kombinasi keduanya:

- Pola yang lebih sederhana: wilayah memanggil wilayah. Seringkali ini merupakan pilihan terbaik untuk layanan eksternal, dan juga sesuai untuk hampir semua layanan internal. Misalnya, saat membangun layanan aplikasi tingkat tinggi di AWS, seperti Amazon API Gateway dan teknologi tanpa server AWS, kami menggunakan pola ini untuk memberikan ketersediaan tinggi, bahkan saat menemui ketidaksesuaikan Availability Zone.

- Pola yang lebih rumit: wilayah memanggil zona atau zona memanggil zona. Saat mendesain komponen bidang data internal, dan eksternal di beberapa kasus, dengan Amazon EC2 (contohnya, peralatan jaringan atau infrastruktur lain yang secara langsung berada di jalur data penting) kami mengikuti pola independensi Availability Zone dan menggunakan instans yang diisolasi di Availability Zone, sehingga lalu lintas jaringan tetap berada di dalam Availability Zone yang sama. Pola ini tidak hanya membantu ketidaksesuaian terisolasi di dalam Availability Zone, namun juga memiliki karakteristik biaya lalu lintas jaringan yang menguntungkan di AWS.

Penutup

Dalam artikel ini, kita telah mendiskusikan beberapa strategi sederhana yang kami gunakan di AWS agar berhasil mengambil dependensi di Availability Zone. Kita telah belajar bahwa kunci bagi stabilitas statis adalah dengan mengantisipasi ketidaksesuaian sebelum itu terjadi. Baik sistem berjalan di armada aktif-aktif yang dapat diskalakan secara horizontal, atau itu merupakan pola aktif-siaga berstatus penuh, kita dapat menggunakan Availability Zone untuk menargetkan ketersediaan tingkat tinggi. Kita menerapkan sistem agar seluruh kapasitas yang akan diperlukan pada kejadian ketidaksesuaian telah diesediakan secara penuh dan siap dijalankan. Terakhir, kita melihat lebih dalam tentang bagaimana Amazon EC2 menggunakan konsep stabilitas statis untuk menjaga indepensi Availability Zone.