Google Cloud

Laporan Resmi Google Cloud bulan November 2019

Alasan Pentingnya Keamanan Container bagi Bisnis Anda

Memahami konsep keamanan container yang memengaruhi organisasi Anda

Google Cloud

Isi

Pengantar
Alasan pentingnya keamanan container bagi Bisnis Anda

Baik Anda sadari atau tidak, semua aplikasi di sekitar Anda berada dalam container. Berkat aplikasi ini, Anda bisa menikmati koneksi Wi-Fi di kafe setempat, memproses pembelian di toko kelontong, dan menggunakan perbankan online. Anda bisa mengirim pesan ke dokter melalui aplikasi seluler, lalu bermain game seluler favorit Anda, semua berjalan dalam container.

Makin besar beban kerja, makin besar konsekuensinya. Bisnis Anda adalah data Anda, dan menjalankan data yang penting bagi bisnis dalam container telah mengubah sesuatu yang dahulu hangat dibicarakan orang menjadi sesuatu yang krusial bagi para pemimpin bisnis. Ketika sedang mengevaluasi penyedia cloud dan pimpinan keamanan Anda berkata, “Cloud A menawarkan resource image terkelola; sedangkan jika menggunakan Cloud B, kita harus mengurusnya sendiri,” seberapa pentingkah hal itu? Apakah hal itu akan memengaruhi keputusan Anda?

Buku ini dimaksudkan untuk menyajikan dasar-dasar keamanan container agar, ketika perlu mengambil keputusan bisnis, Anda mengetahui konteks yang dibutuhkan demi mengamankan bisnis Anda.

Sebenarnya, apa yang dimaksud dengan container?

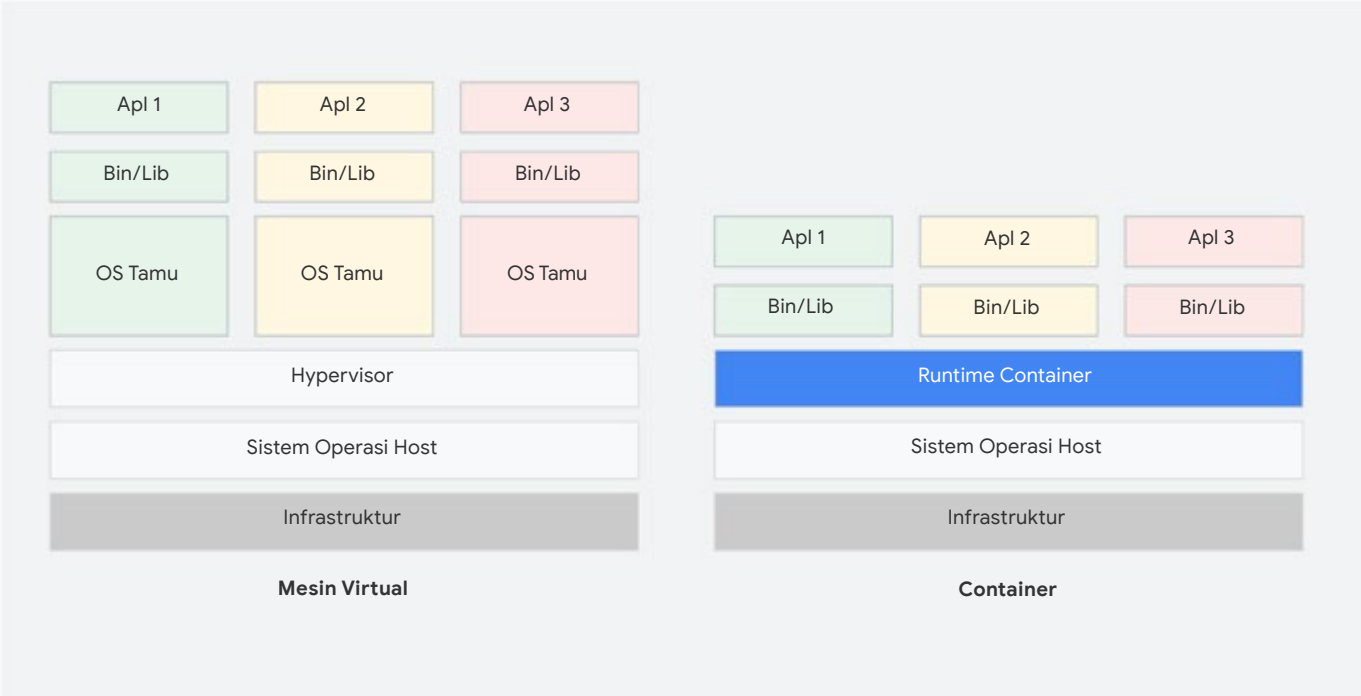
Container adalah salah satu metode pengemasan kode dan dependensi aplikasi agar aplikasi tersebut berjalan lancar di semua lingkungan komputasi. Dengan metode ini, masalah umum terkait portabilitas, atau lebih tepatnya kurangnya portabilitas, dapat dipecahkan. Aplikasi dibuat dan diuji menggunakan versi bahasa, runtime, paket, dan library tertentu. Saat Developer A menyerahkan project kepada Developer B, yang menggabungkannya untuk diuji dan diproduksi, inkonsistensi antara lingkungan tersebut dapat membuat aplikasinya rusak. (Sebagai contoh, versi sistem operasi mungkin sulit diselaraskan dalam tahap pengembangan dan produksi; upgrade OS dan aplikasi berisiko menimbulkan perubahan yang tidak kompatibel secara tidak sengaja.) Selain itu, jika Anda ingin menjalankan beberapa aplikasi di host yang sama, aplikasi tersebut mungkin memerlukan versi OS yang tidak kompatibel.

Container menyelesaikan masalah portabilitas ini dengan mengisolasi aplikasi dari dependensinya, sehingga pemindahan aplikasi antarmesin dapat berjalan lancar. Proses yang berjalan dalam container terisolasi dari lingkungan dasarnya. Anda mengontrol apa yang dapat dilihat dan resource yang dapat diakses oleh container. Dengan begitu, penggunaan resource jadi lebih efisien dan Anda tidak perlu memikirkan infrastruktur dasarnya.

Namun, meski container bisa dianggap sebagai suatu batas, tetapi batas tersebut juga memiliki keterbatasan. Layaknya VM, container masih dapat disusupi melalui berbagai serangan, atau menjadi rentan akibat kesalahan konfigurasi atau komponen yang tidak diperbaiki. Di bagian “Menjalankan Kubernetes dengan Aman”, kami akan menjelaskan lebih lanjut ancaman terhadap container. Namun, beban kerja dan resource komputasi Anda akan rentan terhadap akses yang tidak diizinkan jika container-nya telah disusupi atau memiliki kesalahan konfigurasi, dan bahkan ada kemungkinan aplikasi Anda (dan datanya) dibuat ulang di tempat lain.

Ringkasan

- Jangan menyalahartikan image container; container bukanlah batas keamanan khusus
- Container menggunakan sistem dasar kernel Linux (cgroup, namespace) untuk mengisolasi proses dalam suatu lingkungan
- “Image container” adalah untuk aplikasi Anda dan dependensinya, dan menggunakan “image dasar” sebagai dasarnya.
- Registry container menghosting image container Anda. Anda harus dapat memercayai image dasar dan image container, serta menggunakan registry pribadi dan tepercaya.



Container

Mekanisme container kernel inti dan pembatasan hak istimewa

Container menggunakan fitur khusus kernel Linux yang "mengelabui" aplikasi tertentu sehingga menganggap sedang berada di lingkungan uniknya, meskipun beberapa aplikasi berbagi kernel host yang sama. (Jika Anda belum mengetahuinya, kernel Linux adalah bagian dari sistem operasi yang berkomunikasi di antara proses--permintaan yang menjalankan tugas pengguna, seperti membuka file dan menjalankan program--dan hardware. Kernel Linux mengelola resource seperti memori dan CPU untuk memenuhi permintaan tersebut).

Komponen inti dari kernel Linux yang digunakan untuk container adalah **cgroup** — grup kontrol, yang menentukan resource seperti CPU dan memori yang tersedia untuk proses tertentu — dan **namespace**, yang merupakan cara untuk memisahkan proses dengan membatasi apa yang dapat dilihat oleh setiap proses, sehingga resource sistem "tampak" terpisah untuk proses tersebut.

Untuk mengonfigurasi kapabilitas container, Anda tidak hanya dapat menggunakan cgroup dan namespace, tetapi juga Modul Keamanan Linux (LSM). Dua LSM yang biasa digunakan dalam container adalah AppArmor dan SELinux. Keduanya akan menolak kapabilitas default yang tidak diinginkan, misalnya kemampuan untuk menulis ke sistem file proc. Fitur kernel lainnya, Komputasi Aman dengan filter (**seccomp**) adalah filter panggilan sistem yang mencegah panggilan sistem tertentu ke kernel demi meminimalkan permukaan serangan kernel.

Cgroup, namespace, LSM, dan seccomp adalah parameter yang menentukan apa yang dilakukan oleh suatu proses saat sedang berjalan, dan apa yang menciptakan lingkungan dalam container kita. Namun, container bukan hanya itu saja. Dengan alat ini, kita dapat mengisolasi proses di satu host. Namun, tujuan akhirnya adalah mengemas aplikasi agar mampu berjalan di semua lingkungan dan, untuk itu, kita membutuhkan runtime dan image container.

Runtime, image, dan registry container

Runtime container berfungsi mengeksekusi spesifikasi container. Ian Lewis, Developer Advocate Google Cloud, membahas mekanisme dan fungsionalitas runtime container dalam artikelnya yang terbagi dalam empat bagian. Untuk saat ini, runtime bisa kita anggap sebagai sesuatu yang mengonfigurasi sistem dasar isolasi container dan menjalankan proses di dalamnya. Setiap runtime memiliki kapabilitas keamanannya sendiri-sendiri, terutama di area isolasi container (dijelaskan di bagian "Memahami Isolasi").

Image container menentukan sistem file container. Misalnya, jika Anda menjalankan aplikasi Node.js, image containernya akan berisi aplikasi Anda, Node.js, dan dependesi lain seperti library sistem Linux (kecuali kernelnya). Image container biasanya memperluas image sistem operasi dasar, atau **image dasar**. Karena menjadi dasar bagi container Anda, sebaiknya pastikan image dasar ini sudah di-patch dengan tepat dan bebas dari kerentanan yang diketahui.

Image container bersifat statis, yang menjadikannya salah satu alat keamanan; jika perlu mengubah container yang di-deploy, sebaiknya buat dan deploy image baru, bukan mengubah container yang sedang berjalan. Salah satu cara memanfaatkan properti arsitektur yang ada dalam container sebagai alat keamanan adalah dengan men-deploy container Anda menggunakan sistem file hanya-baca agar penyusup tidak bisa mengubah file.

Image container tersimpan di **registry container**. Anda dapat mengambil image container dari repositori umum, seperti Docker Hub, atau repositori pribadi, seperti Google Container Registry. Apa pun itu, Anda harus dapat memercayai image ini karena image ini berjalan di lingkungan Anda. Mudah dibayangkan betapa berisikonya mengambil

Image container biasanya memperluas image sistem operasi dasar, atau **image dasar.** Karena menjadi dasar bagi container Anda, sebaiknya pastikan image dasar ini sudah di-patch dengan tepat dan bebas dari kerentanan yang diketahui.

image dari sumber yang tidak dikenal tetapi tersedia untuk umum di internet publik (kita akan membahasnya lebih lanjut di bagian “Keamanan Supply Chain”).

Anda tidak perlu memahami cara mengonfigurasi SELinux dan namespace. Namun, Anda perlu tahu bahwa container sendiri bukanlah batas keamanan yang telah melalui proses hardening. Ada banyak komponen penting untuk menjalankannya, misalnya image dasar, image container, dan registry, yang masing-masing memiliki serangkaian pertimbangan keamanannya sendiri.

Keamanan Supply Chain

Menerapkan container dan alat orkestrasi container, seperti Kubernetes, mungkin terdengar menakutkan. Namun, kenyataannya, Anda dapat menggunakan container untuk meningkatkan kualitas keamanan secara keseluruhan.

Anda dapat menggunakan container untuk meningkatkan kualitas keamanan secara keseluruhan:

1. Container memiliki masa aktif yang singkat dan sering di-deploy ulang; **Anda dapat terus melakukan patching**.
2. Container sengaja disetel agar tidak dapat diubah; **jika container berubah, berarti keamanan Anda telah disusupi**.
3. Setelan default keamanan yang baik adalah dengan mengubah satu baris; **menyetel konfigurasi yang aman itu mudah**.
4. Dengan teknologi isolasi, **Anda dapat meningkatkan keamanan tanpa perlu menambahkan resource**.

Container memberi Anda supply chain software

Pada aplikasi monolitik yang berjalan di mesin virtual, developer biasanya melakukan perubahan dengan login ke perangkat dari jarak jauh, atau menerapkan perubahan kode secara manual. Proses ini tidak hanya sulit di-debug, tetapi juga sangat informal; jika nantinya perlu melakukan perubahan, developer cukup login ke VM lagi dari jarak jauh untuk mendebug, mem-patch, mengupdate, memulai ulang, atau menyesuaikan aplikasinya. Proses itu memang kurang aman dan menyulitkan tim operasional karena mereka tidak tahu lagi proses apa yang sedang berjalan.

Namun, lain ceritanya jika mereka menggunakan container. Container memiliki pipeline pengembangan khusus, yang disebut juga supply chain software. Anda dapat menulis kode dan memastikan bahwa kode itu memenuhi persyaratan pembuatan, pengujian, pemindaian, dan lain-lain, sebelum di-deploy. Selain itu, kode yang tidak memenuhi persyaratan dapat dicegat di tahap apa pun dalam supply chain.

Ringkasan

Container memiliki properti yang bermanfaat bagi keamanan:

- Container memiliki pipeline pengembangan khusus
- Container tidak di-patch secara langsung; patch dapat diluncurkan sebagai bagian dari pipeline reguler Anda
- Image container disetel agar tidak dapat diubah; jika ada kerentanan baru, berarti image tersebut telah disusupi

Container memungkinkan Anda melakukan patching secara kontinu dan otomatis

Bahkan saat ini, banyak serangan keamanan yang terjadi di sana-sini, terutama terhadap container, merupakan serangan 'tembak lari' di mana pelaku mencari deployment yang terlihat rentan untuk dieksploitasi. Selain itu, kerentanan tersebut sering kali bukanlah kerentanan yang baru saja diketahui dan belum di-patch, melainkan kerentanan yang diabaikan sejak lama. Layaknya mengoleskan tabir surya ke tubuh, memindai dan mem-patch kerentanan adalah salah satu praktik terbaik yang membosankan tetapi harus dilakukan (maukah Anda menggunakan Pemindai Kerentanan Container Registry?).

Namun, mem-patch container itu berbeda dengan mem-patch VM. Container disetel agar tidak bisa diubah. Artinya, container tidak akan berubah begitu di-deploy; Anda harus membuat dan men-deploy ulang keseluruhan image, bukannya login ke mesin dari jarak jauh. Hal ini sering terjadi, karena container memiliki masa aktif yang singkat; Sysdig memperkiraan bahwa 95% container memiliki masa aktif kurang dari seminggu.

Tapi tunggu dulu...apakah benar sering terjadi? Jika Anda melihat pengelolaan patch tradisional, Patch Tuesday muncul hanya sebulan sekali. Mungkin jika sedang sangat sibuk, Anda juga perlu mengelola beberapa kumpulan patch mingguan. Anda mungkin masih perlu melakukan pemeliharaan pada hari Minggu jam 2 dini hari untuk menerapkan patch (kasihan sekali mereka yang harus begadang untuk melakukannya), tetapi memang kita tidak memiliki banyak waktu untuk mengurusi deployment yang hanya aktif selama seminggu.

Intinya begini. Ketika menggunakan container, Anda tidak mem-patch container aktif, melainkan image dalam registry container Anda. Dengan begitu, image container yang sudah di-patch sepenuhnya dapat diluncurkan atau di-roll back sebagai satu unit, agar proses peluncuran patch sama dengan proses peluncuran kode (yang sering dilakukan), lengkap dengan monitoring, peluncuran terbatas, dan pengujian. Dengan cara ini, patch Anda akan diluncurkan menggunakan proses yang normal dan dapat diprediksi. Cara lainnya (meski kurang disukai karena jadwalnya sulit diprediksi) adalah mengizinkan peluncuran berlangsung secara ad hoc. Lalu, ketika container Anda nonaktif, Kubernetes akan mengaktifkan container lain untuk menggantikannya, dan semua patch yang telah Anda terapkan akan otomatis diluncurkan ke infrastruktur Anda. Bergantung masa aktif container, sebaiknya lakukan patch menyeluruh dalam hitungan hari.

Bahkan saat ini, banyak serangan keamanan yang terjadi di sana-sini, terutama terhadap container, merupakan serangan 'tembak lari' di mana pelaku mencari deployment yang terlihat rentan untuk dieksploitasi.

Dengan container, Anda dapat mengetahui apakah Anda terpengaruh oleh kerentanan baru

Karena sifatnya yang tidak dapat diubah, container memberikan kemudahan dalam melacak konten. Konten disimpan sedemikian rupa sehingga Anda dapat mengambil container berdasarkan kontennya. Dengan begitu Anda benar-benar tahu semua yang berjalan di lingkungan Anda--misalnya, image yang Anda deploy.

Apa artinya ini bagi keamanan? Misalkan, saat memindai image, Anda mendapati bahwa image tersebut sudah di-patch sepenuhnya, sehingga Anda men-deploy-nya. Kemudian, ditemukan kerentanan baru. Bukannya langsung memindai cluster produksi, Anda cukup memeriksa registry untuk mengetahui versi mana yang rentan.

Cara ini juga menyederhanakan pengelolaan patch dengan memisahkan keputusan dan proses terkait kapan melakukan patch dari patching yang sebenarnya. Daripada bertanya “Apakah container saya sudah di-patch?”, tim keamanan Anda dapat bertanya, “Apakah image container saya sudah di-patch?” Lalu, tim operasi Anda dapat bertanya, “Apakah image saya (yang sudah di-patch) berjalan?” Ini juga memungkinkan Anda menjawab pertanyaan tak terelakkan dari CISO: “Apakah kita terdampak?”

Container membuat Google makin aman dan makin andal

Anda tidak perlu memercayai kata-kata kami begitu saja. Infrastruktur Google berada dalam container, berdasarkan sistem orkestrasi container Borg (yang kemudian menjadi Kubernetes), dan kami menggunakannya untuk men-deploy layanan dan patch keamanan di miliaran container setiap minggu.

Sekarang, cara kerjanya tentu sudah cukup jelas: dengan terus mem-patch, dan men-deploy container yang sudah di-patch. Apabila terjadi insiden yang mengganggu, misalnya pemeliharaan hardware atau patch keamanan penting, kami menggunakan proses yang disebut migrasi langsung. Untuk beban kerja GCP, migrasi langsung pada dasarnya adalah blue-green deployment, di mana beban kerja baru di-deploy bersama beban kerja yang ada, lalu load balancer

memindahkan traffic secara bertahap sampai beban kerja tersebut tertangani sepenuhnya oleh instance baru. Artinya, Anda dapat secara efektif mem-patch beban kerja dalam container yang berjalan, tanpa menimbulkan periode nonaktif dan tanpa mengganggu pengguna. Itulah yang membuat kami dapat mem-patch Heartbleed pada tahun 2014 tanpa menimbulkan periode nonaktif, dan yang lebih baru, Spectre/Meltdown.

Singkatnya, container dapat Anda gunakan untuk mem-patch infrastruktur dengan mudah, tanpa menimbulkan periode nonaktif, dan dengan cepat jika Anda terpengaruh oleh kerentanan yang baru ditemukan. Selain itu, Anda dapat mengotomatiskan semua tugas patching yang membosankan dan tidak Anda sukai. Jika Anda mengutamakan keamanan sistem produksi, tim infrastruktur Anda dapat menggunakan container untuk mem-patch lingkungan produksi dengan lebih aman, lebih cepat, dan lebih mudah.

Namun, bagaimana jika Anda ingin menjalankan beberapa container? Gunakan platform orkestrasi container. Selanjutnya, kita akan fokus pada platform utama, **Kubernetes**, yang membantu Anda menjalankan container dalam skala besar.

Menjalankan Kubernetes dengan Aman

Kubernetes upstream, versi open source yang Anda peroleh dari repositori GitHub, tidak didesain untuk menjadi lingkungan keamanan terkunci siap pakai. Namun, setelan defaultnya memecahkan masalah fleksibilitas dan kemudahan penggunaan; versi ini didesain agar mudah diperluas, dan sebagian besar keamanannya mengandalkan penggunaan titik-titik ekstensi ini untuk berintegrasi dengan sistem lain seperti identitas dan otorisasi.

Dan itu tidak apa-apa. Itu berarti Kubernetes dapat menjawab berbagai kasus penggunaan. Namun, itu juga berarti Anda tidak dapat berasumsi bahwa setelan default Kubernetes upstream adalah setelan yang pas untuk Anda. Jika ingin men-deploy Kubernetes dengan fokus pada keamanan, ada sejumlah komponen inti yang perlu dipertimbangkan.

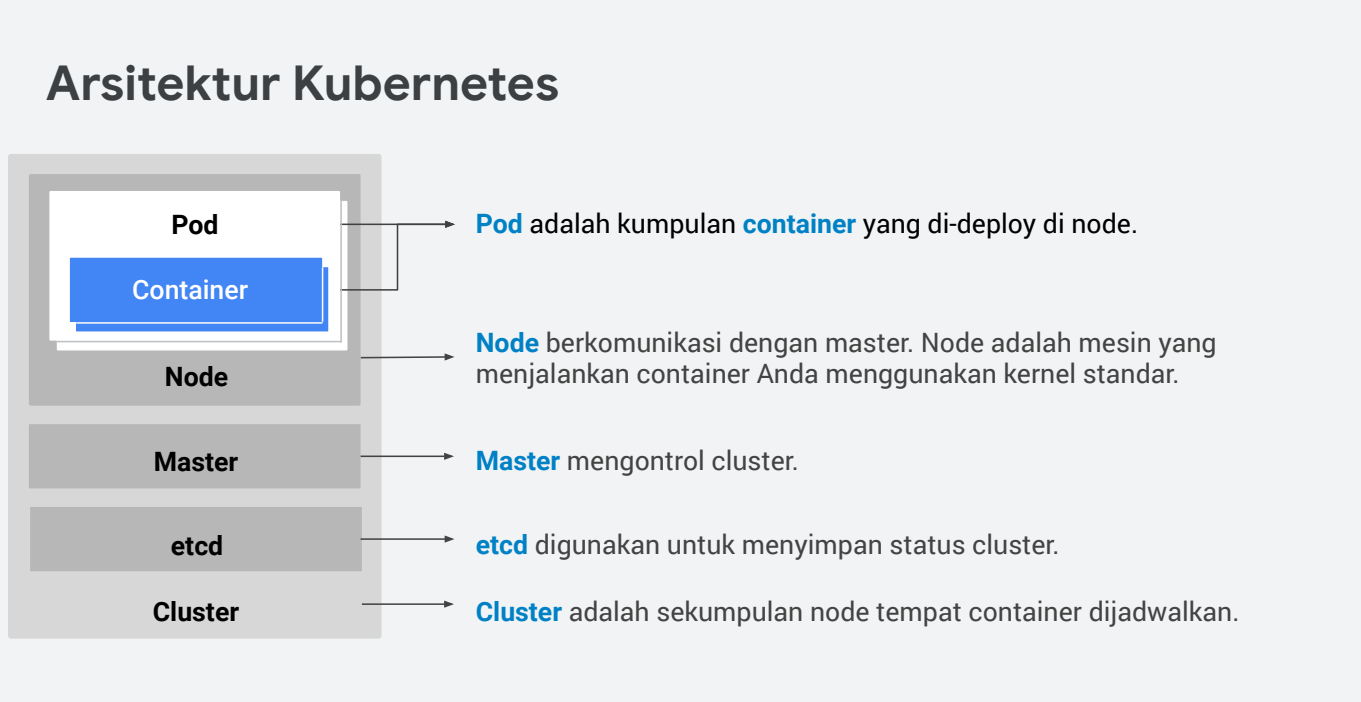
Kita perlu memiliki gambaran mental mengenai Kubernetes terlebih dahulu, sama seperti ketika membahas container. Kubernetes memiliki banyak komponen, dan dari perspektif penyerang, setiap komponen memberikan bonus berbeda jika berhasil disusupi. Dengan memahami hal ini, Anda bisa tahu tindakan yang perlu dilakukan oleh tim keamanan demi melindungi instance Kubernetes dan aplikasi Anda.

Kubernetes dari perspektif penyerang

Di Kubernetes, container berjalan di **pod**. Pod berjalan di **node**, yaitu mesin virtual atau fisik. Node yang menjalankan pod disebut **node pekerja**, yang berisi runtime container, memiliki sistem operasi sendiri, dan dikelola oleh **bidang kontrol Kubernetes**. Terakhir, etcd adalah penyimpanan nilai kunci yang menjaga status bidang kontrol. Semua elemen tersebut membentuk **cluster**.

Ringkasan

- Ada berbagai elemen Kubernetes yang perlu dilindungi
- Alasan penyerangan container antara lain untuk menyalahgunakan resource komputasi, mengakses data beban kerja, atau mengakses kode aplikasi
- Setelan default Kubernetes upstream sebaiknya tidak dianggap mampu memberikan perlindungan memadai berdasarkan kasus penggunaan Anda



Arsitektur Kubernetes

Mengapa penyerang ingin menyusupi komponen ini?

Mari kita mulai dengan container itu sendiri. Alasan utama container diserang saat ini adalah untuk menyalahgunakan resource komputasi, misalnya penambangan mata uang kripto. Namun, penyerangan container juga dapat membuka akses ke data pelanggan atau beban kerja.

Penyerang juga dapat mencoba mengecualikan container untuk mendapatkan akses ke node. Jika node Kubernetes berhasil disusupi, penyerang akan mendapatkan banyak peluang, termasuk kesempatan untuk menyerang node lain di cluster dan juga mendapatkan akses persisten ke kode, komputasi, dan/atau data pengguna yang berharga. “Pengecualian container” adalah suatu jenis serangan eskalasi hak istimewa yang memanfaatkan fakta bahwa container berbagi satu kernel host. Penyerang yang menyusupi container dan memperoleh akses eksklusif kemungkinan bisa mengakses informasi yang berjalan di container lain. Kami akan membahas serangan ini dan cara mencegahnya lebih lanjut di bagian “Memahami Isolasi”.

Master Kubernetes mengontrol cluster Anda. Penyerang yang berhasil menyusupi master dapat mengendalikan lingkungan, termasuk akses untuk membuatnya offline. Sedangkan jika etcd berhasil disusupi, penyerang dapat mengubah dan menghancurkan cluster, mencuri rahasia dan kredensial, atau mendapatkan banyak informasi mengenai aplikasi yang berjalan sehingga bisa membuatnya ulang di tempat lain.

Berita baiknya: Anda dapat secara proaktif melakukan hardening terhadap deployment Kubernetes untuk meningkatkan keamanan container Anda. Jika Anda menggunakan layanan terkelola, penyedia Anda mungkin telah menerapkan beberapa cara tersebut (di Google Kubernetes Engine (GKE), Anda dapat membaca bagian apa yang kami tawarkan untuk melindungi komponen Kubernetes tersebut). Ada juga alat, seperti CIS Benchmark, yang akan membantu Anda membandingkan antara lokasi deployment Kubernetes sekarang dengan lokasi yang Anda inginkan.

CIS Kubernetes Benchmark
Mengukur keamanan Anda

Center for Internet Security (CIS) Kubernetes Benchmark adalah sekumpulan rekomendasi untuk mengonfigurasi Kubernetes guna memperkuat struktur keamanan Anda. CIS Benchmark terkait dengan rilis Kubernetes tertentu. Saat laporan ini ditulis, Benchmark versi pertama digunakan untuk Kubernetes 1.6, sedangkan versi terbarunya digunakan untuk rilis Kubernetes 1.15. CIS Benchmark ini diharapkan dapat digunakan di banyak distribusi Kubernetes.

Setiap CIS Benchmark diperoleh dari kontribusi komunitas dan ditulis oleh pakar dari berbagai bidang guna mengakomodasi beragam perspektif, termasuk konsultasi keamanan, pengembangan, kepatuhan, dan operasi. Para pakar ini bekerja secara cuma-cuma. Jadi, saat Anda bertemu editor CIS, ucapkan terima kasih karena telah mengembangkan resource ini.

Rekomendasi CIS Benchmark terdiri atas beberapa level. Level 1 adalah konfigurasi keamanan fundamental dengan manfaat langsung. Level 2 adalah perluasan dari rekomendasi Level 1 yang, meski memperkuat keamanan, dapat mengganggu performa atau membahayakan kompatibilitas, sehingga perlu dievaluasi sebelum diterapkan.

CIS Kubernetes Benchmark ditulis untuk distribusi Kubernetes open source dan dibuat agar sebisa mungkin dapat diterapkan di semua distribusi secara universal. Jadi, perlu diingat bahwa Benchmark ini bukan petunjuk konfigurasi langkah demi langkah yang bersifat wajib, dan belum bisa diterapkan sepenuhnya ke distribusi yang dihosting, seperti GKE. “Apa pun standar keamanan yang diterapkan, perusahaan harus memikirkan model ancaman mereka,” ujar Rory McCune, Principal Consultant, NCC Group PLC dan editor CIS Benchmark. “Beberapa kontrol keamanan akan selalu memiliki kelebihan dan kekurangan dalam hal dampak terhadap performa dan kemudahan penggunaan. Oleh karena itu, organisasi sebaiknya tidak menganggap metodologi seperti CIS Benchmark sebagai cara wajib. Sebaliknya, organisasi perlu mempertimbangkan setiap rekomendasi dan kecocokannya dalam konteks lingkungan mereka.”

Ringkasan

- CIS Kubernetes Benchmark berisi kumpulan rekomendasi untuk menciptakan struktur keamanan Kubernetes yang kuat
- Benchmark ini bukanlah checklist hal-hal yang wajib dilakukan, melainkan rekomendasi yang perlu Anda evaluasi
- Jika Anda menggunakan layanan terkelola, sejumlah rekomendasi di CIS Benchmark mungkin tidak berlaku

Jika menggunakan layanan terkelola, hanya beberapa item dalam CIS Benchmark yang menjadi tanggung jawab, dapat dilihat, atau dapat dikonfigurasi oleh Anda, karena item tersebut termasuk dalam lingkup tanggung jawab penyedia. Misalnya, dalam GKE, etcd -- penyimpanan nilai kunci yang menjaga status cluster -- adalah bagian dari hardening yang dilakukan Google berdasarkan model tanggung jawab bersama GKE. Jika Anda harus menjalankan alat seperti kube-bench (alat open source yang memeriksa rekomendasi konfigurasi CIS Benchmark), Anda tidak akan dapat memeriksa elemen tertentu, misalnya bidang kontrol, dan mungkin akan melihat item Benchmark yang salah berstatus "FAIL" untuk item lain akibat batasan itu.

Cara menerapkan CIS Benchmark ke deployment Anda

Cara menerapkan CIS Benchmark bergantung pada bagaimana Anda menggunakan Kubernetes dan CIS Benchmark lain apa yang akan digunakan.

Jika Anda menjalankan Kubernetes open source

Jika Kubernetes dijalankan langsung dari upstream, Anda dapat melihat perfoma Anda terhadap CIS Benchmark menggunakan kube-bench, daftar rekomendasi baris demi baris, beserta status PASS/FAIL. Outputnya menampilkan nomor rekomendasi yang terkait dalam panduan Benchmark. Misalnya, jika item 1.1.7 gagal, Anda cukup memeriksa nomor 1.1.7 dalam panduan untuk mengetahui rekomendasi dan implementasinya.

"Kube-bench tidak bisa digunakan untuk memeriksa node master di cluster terkelola, misalnya GKE, EKS, dan AKS, karena tidak ada seorang pun yang bisa mengakses node tersebut. Namun, kube-bench masih bisa digunakan untuk memeriksa konfigurasi node pekerja di lingkungan tersebut."

Jika Anda menggunakan layanan terkelola

Seperti disebutkan dalam dokumen kube-bench, “Kube-bench tidak bisa digunakan untuk memeriksa node master di cluster terkelola, misalnya GKE, EKS, dan AKS, karena tidak ada seorang pun yang bisa mengakses node tersebut. Namun, kube-bench masih bisa digunakan untuk memeriksa konfigurasi node pekerja di lingkungan tersebut.”

Salah satu keuntungan layanan terkelola adalah, bergantung model tanggung jawab bersama dari penyedia Anda, keamanan komponen tertentu (node master, misalnya) tidak menjadi tanggung jawab Anda. Anda masih bisa menggunakan CIS Benchmark dan kube-bench untuk menguji struktur keamanan Anda. Namun, karena adanya pembatasan pemeriksaan, Anda tidak akan melihat status “all pass” (semua berhasil). Untuk komponen yang tidak Anda kontrol (dan mungkin tidak dapat diuji secara langsung), penyedia layanan dapat memberitahukan langkah yang mereka ambil untuk melakukan hardening terhadap komponen tersebut -- dan tentu saja, langkah yang mereka ambil untuk melindungi beban kerja Anda. (Untuk GKE, baca bagian Keamanan bidang kontrol dan Kepercayaan cluster).

Salah satu keuntungan layanan terkelola adalah, bergantung model tanggung jawab bersama dari penyedia Anda, keamanan komponen tertentu (node master, misalnya) tidak menjadi tanggung jawab Anda.

Untuk komponen yang perlindungannya menjadi tanggung jawab Anda, penyedia dapat menawarkan langkah khusus distribusi yang dapat Anda lakukan untuk meningkatkan keamanan, misalnya Panduan hardening GKE.

Terakhir: ada usaha berkesinambungan untuk mengembangkan benchmark bagi distribusi tertentu berdasarkan CIS Kubernetes Benchmark umum. Jika tertarik, Anda dapat mengikuti perkembangan dan berkontribusi langsung di Alat WorkBench CIS.

Memadukan beberapa CIS Benchmark

Sejumlah alat mencoba menganalisis node Kubernetes terhadap beberapa CIS Benchmark (mis., Linux, Docker, dan Kubernetes), lalu menggabungkan hasilnya. Karena benchmark tersebut tidak dirancang untuk dikombinasikan dan diterapkan di lingkungan Kubernetes, saran yang dihasilkan acap kali membingungkan dan mungkin saling bertolak belakang. Misalnya, tidaklah logis untuk bergantung sepenuhnya pada rekomendasi dari CIS Docker Benchmark, seperti mengonfigurasi

plugin otorisasi Docker, jika Kubernetes akan menangani otorisasi dan penjadwalan container. Faktanya, lebih logis untuk menghapus mayoritas fungsi Docker yang tidak digunakan dan berjalan menggunakan image node container GKE seperti yang direkomendasikan dalam panduan hardening GKE.

Demikian pula, CIS Linux Benchmark didesain untuk lingkungan layanan berbeda yang tidak dapat diterapkan sepenuhnya di Kubernetes dan komputasi cloud ter-hosting. Misalnya, beberapa saran berkutat pada tidak dapat mengubah konfigurasi yang sensitif terhadap keamanan tanpa melakukan reboot. Langkah ini logis diterapkan di server tradisional atau deployment desktop karena umumnya reboot dapat dilihat dan jarang dilakukan. Bandingkan dengan cluster Kubernetes yang mayoritas node-nya bersifat sekali pakai serta dapat dibuat dan dihapus secara rutin sesuai kebutuhan aplikasi. Lingkungan komputasi cloud ter-hosing dapat membekukan konfigurasi tertentu dan membiarkan konfigurasi lainnya terbuka agar dapat diubah oleh pengguna. Dalam beberapa kasus, mungkin akan merepotkan jika penyedia cloud membekukan konfigurasi sementara pelanggan perlu mengubah konfigurasi sesuai kebutuhan keamanan mereka.

Jika Anda menggunakan alat yang menggabungkan beberapa benchmark dengan cara ini, sebaiknya pertimbangkan setiap rekomendasi berdasarkan kelebihannya dan putuskan kelogisan dan kecocokannya dengan lingkungan Anda, bukan menganggapnya sebagai praktik terbaik yang wajib dilakukan.

Model tanggung jawab bersama di GKE

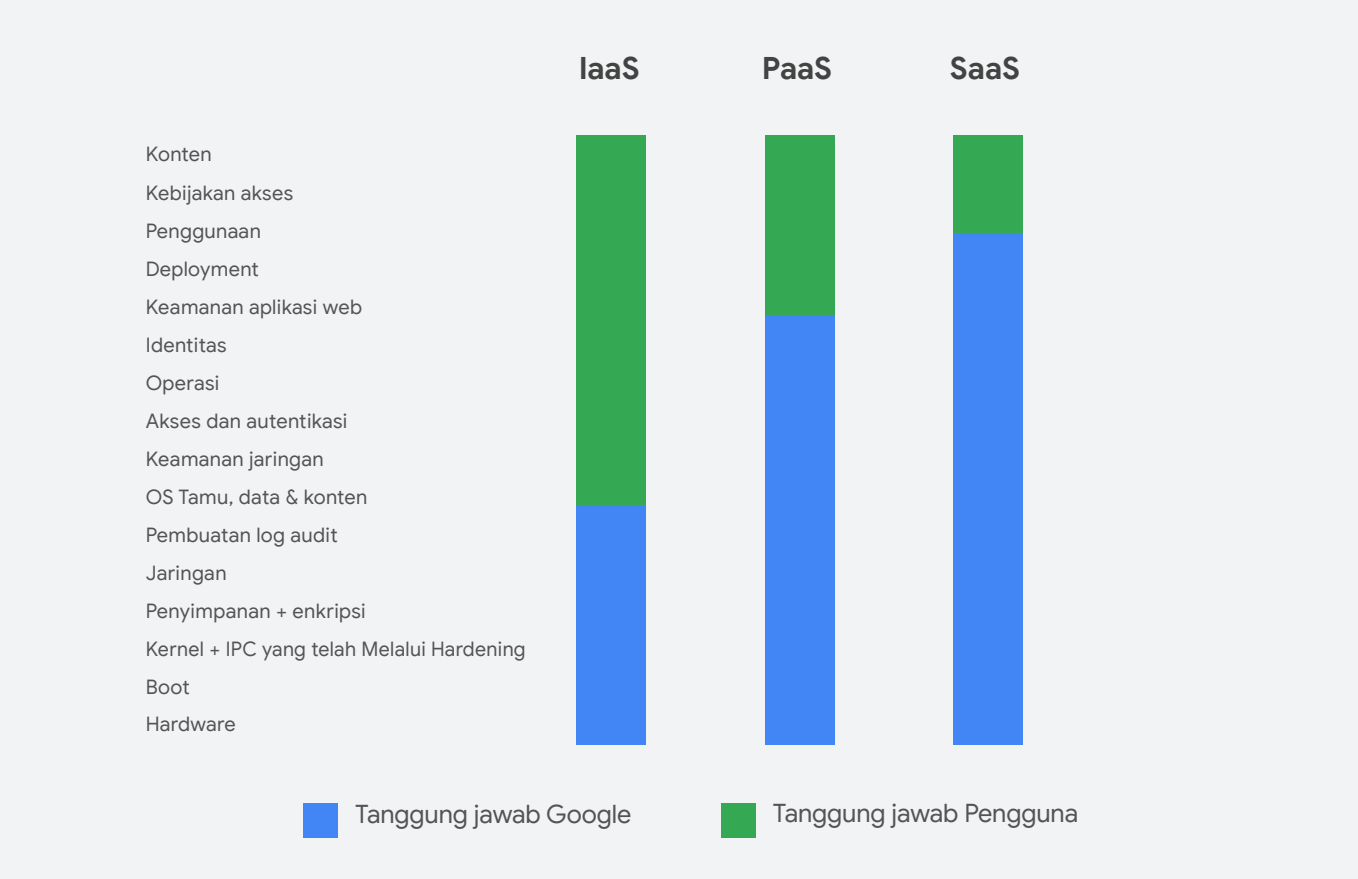
Keamanan di cloud merupakan tanggung jawab bersama antara penyedia cloud dan pelanggannya. Google Cloud berkomitmen untuk melakukan tugasnya dalam melindungi infrastruktur dasar, seperti enkripsi saat data nonaktif secara default, dan menyediakan kapabilitas yang dapat digunakan untuk melindungi beban kerja Anda, seperti kontrol akses di Cloud Identity and Access Management (IAM). Dengan munculnya model infrastruktur yang lebih baru, mengetahui tanggung jawab Anda dan penyedia Anda masing-masing bukanlah hal yang mudah. Di sini, kami akan menjelaskan fungsi Google Kubernetes Engine (GKE) dan tempat memperoleh referensi untuk topik lainnya.

Model tanggung jawab bersama di Google Cloud

Model tanggung jawab bersama tergantung pada beban kerja. Makin banyak yang kita kelola, makin banyak yang dapat kita lindungi. Ini dimulai dari bagian bawah stack ke atas, dari lapisan infrastructure as a service (IaaS), yang hardware, penyimpanan, dan jaringannya menjadi tanggung jawab penyedia, hingga software as a service (SaaS), tempat segala hal (kecuali konten dan aksesnya) bergantung pada penyedia. (Untuk mendapatkan informasi selengkapnya, baca laporan resmi Ringkasan Desain Keamanan Infrastruktur Google). Lapisan platform as a service (PaaS), seperti GKE, berada di tengah-tengah; karenanya timbul ketidakjelasan.

Ringkasan

- Keamanan layanan cloud adalah tanggung jawab bersama antara penyedia dan pengguna
- Tim keamanan dan tanggap insiden harus paham bahwa mereka bertanggung jawab untuk melakukan hardening dan memberikan perlindungan
- Tim Anda harus memahami cara penyedia berkomunikasi apabila terjadi insiden yang memengaruhi komponen yang menjadi tanggung jawab mereka



Tanggung jawab bersama di Google Cloud

Untuk GKE, di level tinggi, kami bertanggung jawab melindungi:

- Infrastruktur dasar, termasuk hardware, firmware, kernel, OS, penyimpanan, jaringan, dan banyak lagi. Ini termasuk mengenkripsi data saat nonaktif secara default, mengenkripsi data saat transit, menggunakan hardware yang didesain khusus, memasang kabel jaringan pribadi, melindungi pusat data dari akses fisik, dan mengikuti praktik pengembangan software yang aman.

- Sistem operasi node, misalnya Container-Optimized OS (COS) atau Ubuntu. GKE akan segera meluncurkan semua patch untuk image ini. Jika auto-upgrade diaktifkan, proses ini akan berjalan otomatis. Ini adalah lapisan dasar container Anda, yang berbeda dengan sistem operasi yang berjalan di container Anda.

- Distribusi Kubernetes. GKE menyediakan versi upstream terbaru untuk Kubernetes dan mendukung sejumlah versi minor. Menyediakan update untuk versi-versi tersebut, termasuk patch, merupakan tanggung jawab kami.

- Bidang kontrol. Di GKE, kami mengelola bidang kontrol, yang mencakup VM master, server API dan komponen lain yang berjalan di VM tersebut, serta database etcd. Ini mencakup upgrade dan patching, penskalaan, dan perbaikan, semua yang didukung oleh SLO.

- Integrasi Google Cloud, untuk IAM, Cloud Audit Logging, Stackdriver, Cloud Key Management Service, Cloud Security Command Center, dll. Produk Cloud tersebut juga menyediakan kontrol untuk beban kerja IaaS di seluruh Google Cloud di GKE.

Sekarang, Anda bertanggung jawab melindungi komponen berikut:

- Node yang menjalankan beban kerja Anda. Anda bertanggung jawab atas software tambahan apa pun yang diinstal di node, atau perubahan konfigurasi yang dilakukan terhadap setelan defaultnya. Anda juga bertanggung jawab untuk selalu mengupdate node. Kami selalu menyediakan image dan konfigurasi VM yang telah melalui hardening, mengelola container yang dibutuhkan untuk menjalankan GKE, dan menyediakan patch untuk OS Anda. Tanggung jawab Anda hanyalah melakukan upgrade.
  Jika Anda menggunakan auto-upgrade node, kamilah yang akan mengupgrade node itu.

- Beban kerja, termasuk kode aplikasi, dockerfile, image container, data, kebijakan RBAC/IAM, serta container dan pod yang Anda jalankan. Ini berarti memanfaatkan fitur GKE dan produk Google Cloud lainnya untuk membantu melindungi container Anda.

Hardening bidang kontrol merupakan tanggung jawab Google

Google bertanggung jawab untuk meningkatkan keamanan bidang kontrol, yaitu komponen yang mengelola cara Kubernetes berkomunikasi dengan cluster, dan menerapkan status yang diinginkan oleh pengguna. Bidang kontrol mencakup VM master, server API, scheduler, pengelola

pengontrol, CA cluster, materi kunci root-of-trust, pengautentikasi dan pemberi izin IAM, konfigurasi logging audit, etcd, dan berbagai pengontrol lainnya. Semua komponen bidang kontrol Anda berjalan di instance Compute Engine yang kami miliki dan operasikan. Instance ini bersifat tenant tunggal. Artinya, setiap instance menjalankan bidang kontrol dan komponennya untuk satu pelanggan saja. (Anda dapat mempelajari keamanan bidang kontrol GKE lebih lanjut di sini.)

Kami melakukan perubahan terhadap bidang kontrol untuk menyempurnakan hardening komponen ini secara rutin, karena serangan dapat terjadi di mana saja saat kerentanan baru diumumkan atau patch baru tersedia. Misalnya, kami mengupdate cluster agar menggunakan RBAC, bukan ABAC secara default dan terkunci, serta akhirnya menonaktifkan dasbor Kubernetes.

Cara menangani kerentanan bergantung pada jenis komponen tempat kerentanan ditemukan:

- Kernel atau sistem operasi: Kami menerapkan patch ke komponen yang terpengaruh, termasuk memperoleh dan menerapkan patch ke image host untuk Kubernetes, COS, dan Ubuntu. Kami otomatis mengupgrade VM master, tetapi Anda bertanggung jawab atas upgrade node. Spectre/Meltdown dan L1TF adalah beberapa contoh kerentanan tersebut.

- Kubernetes: Bersama Googler di Tim Keamanan Produk Kubernetes, kami sering membantu dalam pengembangan dan pengujian patch untuk mengetahui kerentanan Kubernetes saat kerentanan tersebut ditemukan. Karena GKE merupakan distribusi resmi, kami menerima patch sebagai bagian dari Daftar Distributor Pribadi. Kami bertanggung jawab atas peluncuran perubahan terhadap VM master, sedangkan Anda bertanggung jawab atas upgrade node Anda. Baca buletin keamanan berikut untuk mengetahui contoh-contoh terbaru terkait kerentanan tersebut, CVE-2017-1002101, CVE-2017-1002102, dan CVE-2018-1002105.

- Komponen yang digunakan di konfigurasi default Kubernetes Engine, seperti komponen Calico untuk Kebijakan Jaringan, atau etcd. Kami tidak mengontrol project open source yang digunakan di GKE, tetapi kami memilih project open-source yang terbukti memberikan praktik keamanan yang tangguh dan menangani keamanan dengan sungguh-sungguh. Untuk project ini, kami mungkin menerima patch dari Kubernetes upstream, partner, atau daftar distributor untuk project open source lainnya. Kami bertanggung jawab meluncurkan perubahan ini, dan/atau memberi tahu Anda jika ada tindakan yang diperlukan. TTA-2018-001 adalah salah satu contoh kerentanan yang kami patch secara otomatis.

- GKE: Jika ditemukan kerentanan di GKE, misalnya melalui Vulnerability Reward Program, kami wajib mengembangkan dan menerapkan perbaikannya.

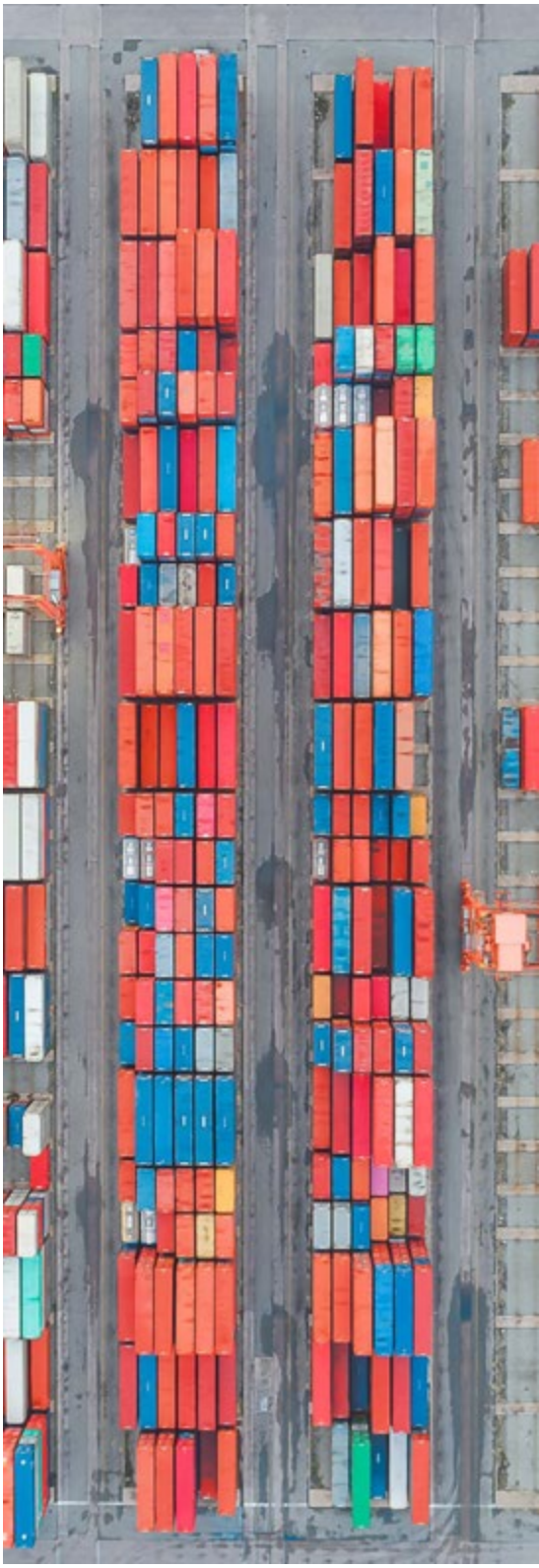
Dalam semua kasus ini, kami menyediakan semua patch tersebut sebagai bagian dari rilis GKE umum (rilis patch dan perbaikan bug) sesegera mungkin dengan mempertimbangkan level risiko, waktu embargo, dan faktor kontekstual lainnya.

Kamilah yang akan bekerja melindungi node, tetapi Andalah yang bertanggung jawab untuk mengupgradenya, lalu menikmati manfaatnya

Node pekerja di Kubernetes Engine terdiri atas beberapa jenis permukaan yang perlu dilindungi, termasuk OS node, runtime container, komponen Kubernetes (misalnya kubelet dan kube-proxy), dan container sistem Google untuk monitoring dan logging. Kami bertanggung jawab mengembangkan dan merilis patch untuk komponen tersebut, sedangkan Anda bertanggung jawab mengupgrade sistem untuk menerapkan patch tersebut.

Komponen Kubernetes seperti kube-proxy dan kube-dns, dan add-on khusus Google untuk menyediakan layanan logging, monitoring, dan layanan lainnya, berjalan di container berbeda. Kami bertanggung jawab atas kompatibilitas, skalabilitas, pengujian upgrade, dan konfigurasi keamanan bidang kontrol container tersebut. Jika komponen tersebut perlu di-patch, Andalah yang harus menerapkan patch tersebut.

Untuk mempermudah deployment patch, Anda dapat menggunakan auto-upgrade node. Auto-upgrade node menerapkan update ke node secara rutin, termasuk update untuk sistem operasi dan komponen Kubernetes dari versi stabil terbaru. Hal ini termasuk patch keamanan. Jika Anda menggunakan auto-upgrade node, Google yang akan mengupgrade node Anda. Khususnya, jika patch berisi perbaikan penting dan bisa diluncurkan sebelum kerentanan diumumkan kepada publik tanpa melanggar embargo, lingkungan GKE Anda akan diupgrade sebelum kerentanan tersebut diumumkan.



Melindungi beban kerja tetap menjadi tanggung jawab Anda

Sejauh ini kita telah membahas infrastruktur dasar yang menjalankan beban kerja. Namun, tentu saja Anda tetap bertanggung jawab atas keamanan aplikasi dan perlindungan lain bagi beban kerja itu sendiri.

Anda juga bertanggung jawab atas konfigurasi Kubernetes yang terkait dengan beban kerja Anda. Tanggung jawab ini mencakup penyiapan NetworkPolicy untuk membatasi traffic antarpod dan penggunaan PodSecurityPolicy untuk membatasi kapabilitas pod. Untuk melihat daftar terbaru praktik terbaik yang kami rekomendasikan untuk melindungi cluster Anda, termasuk konfigurasi node, baca Hardening keamanan cluster Anda.

Jika ada kerentanan dalam image container atau aplikasi, Anda juga bertanggung jawab untuk menerapkan patch-nya. Namun, ada beberapa alat yang bisa membantu Anda:

- Image dasar yang dikelola Google, yang di-patch secara rutin untuk menangani kerentanan yang diketahui.

- Pemindaian kerentanan Container Registry untuk menganalisis potensi kerentanan yang diketahui dalam image dan paket container Anda.

- Cloud Security Scanner untuk membantu Anda mendeteksi kerentanan aplikasi umum.

Tanggap insiden di GKE

Bagaimana reaksi Anda jika cluster Anda diserang meskipun kita sudah sama-sama melakukan tugas kita?

Saran pertama kami: jangan panik. Google Cloud menjaga keamanan infrastruktur kita, termasuk lokasi beban kerja pengguna, dengan sungguh-sungguh. Kami juga telah mendokumentasikan proses tanggap insiden. Tugas tim keamanan kami adalah melindungi Google Cloud dari potensi serangan dan melindungi komponen yang disebutkan di atas. Terkait hal-hal yang menjadi tanggung jawab Anda, Google Cloud memiliki beragam partner keamanan container yang terintegrasi dengan Cloud Security Command Center. Peringatan dan remediasi yang Anda terima dari Cloud Security Command Center dan integrasi partnernya bisa membantu Anda menanggapi masalah yang terjadi di area yang perlindungannya menjadi tanggung jawab Anda.

Saat menanggapi suatu insiden, Anda bisa memanfaatkan Stackdriver Incident Response and Management (alfa) untuk membantu mengurangi waktu mitigasi insiden, melihat contoh kueri untuk log audit Kubernetes, dan menonton presentasi Cloud Forensics 101 dari Next '18 untuk mempelajari cara melakukan forensik lebih lanjut.

Apa poin-poin utama dalam keamanan GKE? Untuk GKE, kami bertanggung jawab melindungi bidang kontrol, yang mencakup VM master, etcd, dan pengontrol. Anda bertanggung jawab melindungi node pekerja, termasuk men-deploy patch ke OS, runtime, dan komponen Kubernentes, serta, tentu saja, mengamankan beban kerja Anda.

Cara mudah untuk melakukan tugas Anda adalah dengan:

Google Cloud menjaga keamanan infrastruktur kita, termasuk lokasi beban kerja pengguna, dengan sungguh-sungguh. Kami juga telah mendokumentasikan proses tanggap insiden.

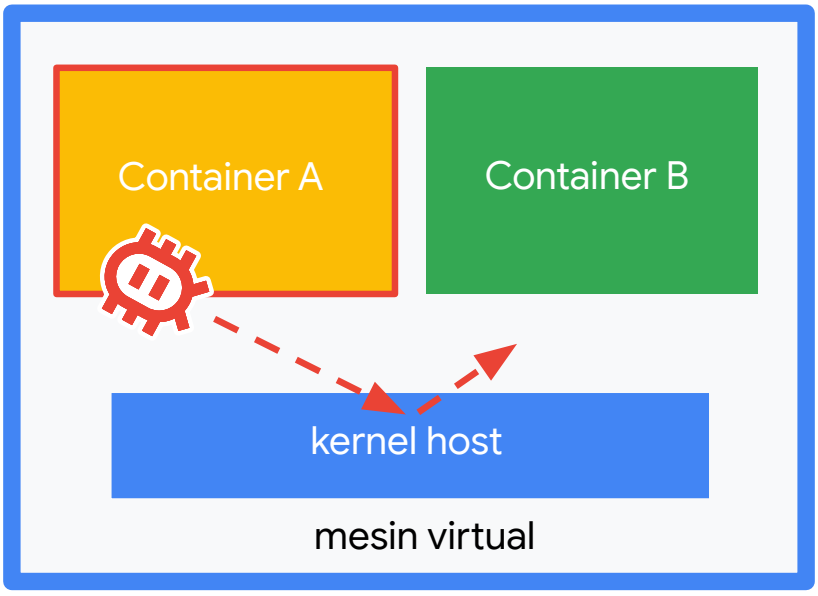
1. Menggunakan auto-upgrade node
2. Melindungi beban kerja dari kerentanan image dan aplikasi yang umum, dan
3. Mengikuti panduan hardening Google Kubernetes Engine.

Dengan mengikuti ketiga langkah di atas, bersama-sama kita dapat membangun lingkungan GKE yang tangguh terhadap serangan dan kerentanan, guna menghadirkan uptime dan performa yang prima.

Memahami Isolasi Container

Salah satu alasan utama penggunaan container adalah agar aplikasi Anda dapat dipisahkan dari lingkungan dasarnya dan mendukung pemanfaatan resource yang lebih tinggi dengan melakukan "pengemasan berkelompok" terhadap beberapa beban kerja ke setiap server. Oleh karena itu, arsitektur container berarti bahwa beban kerja di-deploy menggunakan beberapa container yang berbagi kernel yang sama.

Sayangnya, meski beban kerja yang berbagi kernel menghasilkan densitas dan efisiensi yang lebih tinggi, tetapi itu juga berarti bahwa satu bug kernel dapat menyusupi seluruh host. Pengecualian container adalah salah satu jenis serangan yang mengikuti pola tertentu: pelaku menyerang satu container, mengeskalasi hak istimewa mereka, mendapatkan akses ke host, lalu ke container kedua dan kontennya.



Container tidak menahan

Ringkasan

- Kernel bersama dalam arsitektur container dapat menimbulkan ancaman serangan "pengecualian container"
- Project open source isolasi yang bermunculan, seperti gVisor dan Kata Containers, menghadirkan pertahanan berlapis untuk mencegah serangan ini
- Pertimbangkan untuk menggunakan versi project yang terkelola jika tim Anda tidak bisa mengelola alat open source secara independen

Pergerakan lateral di antara container tetap menjadi ancaman utama bagi pengguna yang menjalankan beban kerja sensitif, menyediakan layanan SaaS, atau menjalankan kode yang tidak tepercaya. Pengguna ini perlu mengambil keputusan terkait risiko kode mereka dan kekurangan yang bisa mereka toleransi, dalam hal overhead performa dan kompatibilitas, demi menyempurnakan isolasi kode tersebut dan keamanan lingkungannya. (Jadi pilihlah yang terbaik untuk Anda.) Misalnya, kode yang ditulis oleh tim Anda sendiri, yang kontrol keamanan untuk peninjauan kodenya telah Anda terapkan, dan yang mengakses set data publik, risikonya dianggap lebih rendah dibanding kode yang mengambil data sensitif atau diberikan oleh pengguna eksternal, atau keduanya. Dalam kasus yang disebut terakhir ini, organisasi dapat memutuskan untuk mengorbankan performa tertentu demi menyempurnakan pengamanan yang mereka terapkan pada kode itu.

Pergerakan lateral di antara container tetap menjadi ancaman utama bagi pengguna yang menjalankan beban kerja sensitif, menyediakan layanan SaaS, atau menjalankan kode yang tidak tepercaya.

Kasus penggunaan ini telah melahirkan beberapa project open source yang bertujuan meningkatkan isolasi beban kerja container, sekaligus mempertahankan manfaat densitasnya sebaik mungkin. Dalam aneka project yang bermunculan ini, terdapat dua jenis pendekatan: isolasi melalui kernel sekunder yang berjalan di mesin virtual dan isolasi melalui unikernel. Dengan memahami setiap pendekatan dan perbedaan setiap solusi, Anda dapat memilih project isolasi yang tepat untuk mendukung beban kerja Anda.

Pendekatan hypervisor

Hypervisor adalah batas yang paling dipahami, dan secara umum dianggap lebih kuat dibanding batas container. Project yang menggunakan pendekatan ini berhasil menemukan metode untuk menghasilkan mesin virtual ringan yang akan berfungsi sebagai batas sekunder untuk setiap container atau pod, dan untuk setiap container agar memiliki kernelnya sendiri. Pendekatan ini menghadirkan isolasi memori, jaringan, I/O, dan artinya bahwa ada banyak opsi bagi pengguna yang menggunakan teknologi virtualisasi lama.

Salah satu project yang menggunakan pendekatan ini adalah Kata Containers. Arsitektur Kata Containers memiliki beberapa komponen, yang dapat meningkatkan waktu start-up dan kompleksitas. Namun, komponen tersebut sangat kompatibel untuk dipasang di mana pun sehingga pengguna dapat memanfaatkan fleksibilitasnya. Misalnya, pengguna dapat memutuskan untuk menggunakan QEMU, Firecracker, atau Rust VMM sebagai pemantau mesin virtualnya. Karena Kata Containers bergantung pada fitur virtualisasi hardware, pengguna harus menggunakan penyedia layanan cloud yang mendukung virtualisasi bertingkat, menawarkan bare-metal, atau mainframe Intel VTX, Arm HYP, IBM Power, atau IBM Z.

Pendekatan unikernel

Pendekatan unikernel berfokus pada meminimalkan permukaan serangan kernel host dengan menyediakan kernel sederhana yang hanya memiliki fungsi yang dibutuhkan oleh beban kerja dalam container. Keterbatasan fungsi ini membuat penyerang memiliki keterbatasan metode untuk mengakses host, sehingga peluang eksploitasi juga terbatas.

Karena hanya menggunakan sedikit komponen baru di arsitekturnya, pendekatan unikernel bisa mempercepat waktu start-up dan meminimalkan dampak terhadap performa berdasarkan beban kerjanya. Namun, karena pendekatan unikernel memang sengaja membatasi fungsi kernel, maka fungsi tersebut harus kompatibel dengan beban kerja yang diharapkan.

Project seperti gVisor dan Nabla Containers menerapkan pendekatan unikernel untuk melakukan isolasi. gVisor didasarkan pada sistem internal yang digunakan Google untuk mengisolasi container. Selain menggunakan kernel ruang pengguna, gVisor juga menggunakan proxy filesystem agar kernel tersebut dapat mengakses filesystem yang difilter.

Menentukan persyaratan terlebih dahulu

Menentukan solusi isolasi yang sesuai dengan kebutuhan membutuhkan kerja sama antara tim pengembangan dan tim keamanan Anda. Langkah pertamanya adalah menentukan profil risiko dan ancaman di organisasi Anda dan beragam beban kerja dalam container, lalu memutuskan beban kerja mana yang membutuhkan isolasi tambahan. Dari sini, Anda bisa membandingkan persyaratan dan prioritas beban kerja tersebut dengan opsi isolasi yang ada, serta toleransi tim untuk mengelola suatu project secara independen. Jika ternyata Anda membutuhkan versi terkelola, GKE Sandbox dibuat berdasarkan gVisor dan tersedia untuk pengguna GKE. Persyaratan tersebut akan membantu Anda menemukan solusi isolasi yang sesuai dengan kebutuhan aplikasi.

Berkontribusi bagi Kubernetes open source

Sebagai sebuah project open source, siklus proses pengembangan Kubernetes berkutat di antara pengguna dan developer. Pengguna menentukan arah project melalui kebutuhan dan kasus penggunaan mereka, sedangkan developer menyumbangkan kode yang membawa Kubernetes ke iterasi selanjutnya. Sebagai organisasi yang menggunakan Kubernetes, meskipun Anda menggunakannya sebagai layanan terkelola, ada banyak cara untuk menyumbangkan kontribusi Anda bagi project yang penting ini.

Masukan pengguna

Developer sangat membutuhkan masukan dari pengguna. Jika menjalankan Kubernetes dan menghadapi masalah atau ingin meminta fitur baru, Anda dapat menyampaikannya di Kubernetes GitHub dan membagikan apa yang Anda temukan. Demikian juga, jika ada yang berfungsi dengan baik, bagikan agar komunitas mengetahuinya. Anda bisa membagikan masukan Anda kepada komunitas.

Luangkan waktu untuk memberikan kontribusi ke upstream

Begitu tim Anda mulai aktif dan berjalan, sebaiknya luangkan waktu dalam organisasi Anda untuk memberikan kontribusi ke upstream (artinya kembali ke project open source), meskipun Anda menggunakan Kubernetes melalui layanan. Secara khusus, jika tim Anda memiliki keahlian tertentu yang berasal dari kasus penggunaan (misalnya, Anda bekerja di industri yang diatur oleh undang-undang, seperti perbankan, atau menjalankan Kubernetes di lingkungan multi-tenant), berbagi pengetahuan sebagai kontributor akan sangat bermanfaat baik bagi Anda sendiri maupun komunitas luas.

Ringkasan

- Project open source bisa berkembang jika pengguna membagikan kisah, waktu, dan keahliannya bagi project tersebut
- Sebaiknya tim Anda meluangkan waktu untuk memberikan kontribusi ke upstream, baik melalui kode atau interaksi dengan komunitas

Namun, kontribusi Anda tidak harus berupa kode. Ada banyak cara untuk berkontribusi, termasuk dengan dokumentasi, pengelolaan rilis, atau grup kerja untuk arsitektur tertentu (misalnya Grup Kerja Multi-tenancy). Anda dapat mempelajari cara mulai berkontribusi ke upstream dengan membaca Panduan Kontributor Kubernetes.

Mengirimkan studi kasus atau blog

Komunitas selalu ingin membagikan kisah tentang Kubernetes. Jika ingin menceritakan perjalanan migrasi Anda, masalah yang Anda temukan, atau bagaimana Kubernetes telah membantu organisasi Anda, kirimkan cerita Anda ke Blog Kubernetes. Jika memiliki kisah sukses yang lebih besar, Anda bisa mengembangkan studi kasus dengan bantuan Cloud Native Computing Foundation.

Software open source bisa berkembang karena makin banyak orang dan organisasi yang menggunakannya. Usaha bersama ini dan komunitas kolaborasi yang dilahirkannya membuka lebar peluang pembelajaran, pengembangan, dan sosial. Begitu organisasi Anda sudah mulai aktif dan berjalan, pikirkan apa yang dapat Anda lakukan untuk terlibat dalam komunitas project open source. Pendapat dan pengalaman Anda akan menjadikan Kubernetes lebih baik bagi semua orang.

Menggabungkan semuanya

Aplikasi modern berjalan di container; data yang penting bagi perusahaan Anda terikat pada teknologi container. Sebagai pimpinan bisnis, Anda mungkin tidak perlu mengetahui seluk-beluk keamanan Kubernetes. Yang perlu Anda ketahui adalah bahwa Kubernetes siap pakai tidak akan memberikan perlindungan yang Anda butuhkan.

Hal ini tidak hanya berlaku pada container dan Kubernetes; sama seperti teknologi lainnya, Anda harus menentukan persyaratan keamanan tertentu dan berinvestasi untuk mewujudkannya. Untuk container, ini berarti memikirkan model ancaman dan mengeksplorasi solusi di area berikut:

- **Keamanan infrastruktur:** Bagaimana cara melindungi infrastruktur yang menjalankan container? Jika menggunakan layanan container terkelola, apa perlindungan yang menjadi tanggung jawab Anda? Apakah tim Anda memiliki rencana tanggap insiden?

- **Keamanan supply chain:** Bagaimana cara memastikan kepercayaan di sepanjang siklus proses pengembangan, pembuatan, dan deployment? Dari mana image container Anda berasal? Bagaimana cara memverifikasi bahwa Anda memercayai yang Anda deploy?

- **Keamanan runtime:** Aplikasi apa yang membutuhkan keamanan tambahan selain keamanan default organisasi Anda? Bagaimana cara meningkatkan kekuatan pertahanannya dan mengisolasi lebih lanjut aplikasi yang berisiko? Bagaimana cara Anda mengetahui dugaan insiden?

Referensi seperti CIS Benchmark dapat membantu tim pengembangan Anda mempelajari praktik yang direkomendasikan. Dengan meluangkan waktu di komunitas Kubernetes upstream, tim pengembangan Anda akan memperoleh banyak resource dan koneksi yang berharga bagi organisasi, sekaligus mendukung kualitas project Kubernetes secara keseluruhan.

Keamanan adalah upaya tanpa akhir, bukan sekali selesai. Teknologi yang menjalankan aplikasi Anda terus berkembang; oleh karena itu, langkah untuk mengamankan dan melindungi data penting perusahaan Anda juga harus terus ditingkatkan.

Kisah Pengguna: Cara DroneDeploy mendapatkan sertifikasi ISO-27001 untuk GKE

Catatan editor: Perusahaan pemetaan data udara, DroneDeploy, ingin memigrasikan lingkungan Kubernetes lokalnya ke Google Kubernetes Engine, jika auditor mengizinkannya. Lanjutkan membaca untuk mengetahui cara perusahaan ini memanfaatkan kapabilitas keamanan native GKE guna mendapatkan sertifikasi ISO-27001.

Di DroneDeploy, kami bekerja keras untuk mengamankan data pelanggan. Kami selalu bangga akan usaha keamanan internal kami. Menerima sertifikasi kepatuhan adalah bentuk penghargaan atas usaha tersebut, sehingga kami dapat membakukan program keamanan informasi, dan membuat kami terus bekerja dengan standar yang tinggi. Belum lama ini, kami berhasil memperoleh sertifikasi ISO-27001. Semuanya berkat memanfaatkan praktik keamanan yang ada di Google Cloud dan Google Kubernetes Engine (GKE). Berikut cara kami melakukannya.

Sebagai startup B2B SaaS yang berkembang pesat di San Francisco, misi kami adalah membuat data udara dapat diakses dan dimanfaatkan oleh semua orang. Caranya adalah dengan menyediakan pemrosesan citra, pemetaan otomatis, pemodelan 3D, berbagi data, dan kontrol penerbangan bagi pengguna melalui aplikasi iOS dan Android. Platform Enterpise kami menyediakan konsol admin untuk akses berbasis peran dan pemantauan penerbangan, rute yang dipetakan, pengambilan citra, dan berbagi citra. Selain melayani lebih dari 4.000 pelanggan di industri konstruksi, energi, asuransi, dan tambang di 180 negara, kami juga menyerap lebih dari 50 terabyte data citra dari 30.000 lebih penerbangan berbeda setiap bulannya.

Banyak pelanggan dan calon pelanggan kami adalah perusahaan besar yang memiliki ekspektasi keamanan yang ketat terhadap penyedia layanan pihak ketiga mereka. Di era dengan aturan yang semakin ketat (misalnya undang-undang GDPR Eropa) dan masalah keamanan data, perhatian pada pengelolaan keamanan informasi masih rendah. Inisiatif kepatuhan merupakan bagian dari strategi keamanan komprehensif yang membantu kami mengomunikasikan komitmen dalam mengamankan data pelanggan. Di DroneDeploy, kami memulai proses kepatuhan kami dengan ISO-27001, yaitu standar keamanan informasi internasional yang diakui di berbagai industri.

> Kami berhasil memperoleh sertifikasi ISO-27001. Semuanya berkat memanfaatkan praktik keamanan yang ada di Google Cloud dan Google Kubernetes Engine (GKE)."

Arsitektur DroneDeploy: Google Kubernetes Engine (GKE)

DroneDeploy adalah salah satu perusahaan pertama yang menggunakan Kubernetes, dan, sejak saat itu, kami memigrasikan beban kerja kami dari mesin virtual ke container yang diorkestrasi oleh Kubernetes. Kini kami menjalankan lebih dari 150.000 tugas Kubernetes setiap bulannya, dengan waktu proses berkisar dari lima menit hingga beberapa hari. Fitur kami untuk mengelola cluster berkembang seiring berjalannya waktu, mulai dari bash dan skrip Ansible buatan sendiri, hingga kops yang sudah tersebar di mana-mana (dan luar biasa). Sekitar 18 bulan lalu, kami mengevaluasi kembali strategi hosting kami dengan mempertimbangkan penurunan biaya komputasi di cloud. Kami sadar bahwa mengelola cluster Kubernetes sendiri kurang menguntungkan bagi bisnis, dan akan lebih baik jika kami berfokus ke hal lain, bila memungkinkan.

Kami menganalisis penawaran Kubernetes terkelola dari beberapa penyedia cloud terkemuka dan melakukan uji tuntas teknis sebelum memilih, tidak hanya membandingkan semua yang tersedia saat itu, tetapi juga roadmap ke depan. Kami tahu bahwa GKE memiliki beberapa fitur utama yang tidak dimiliki oleh penyedia lain, misalnya penskalaan otomatis Kubernetes-native yang tangguh, bidang kontrol yang canggih, master zona yang memiliki beberapa ketersediaan, dan dokumentasi yang lengkap. Kemampuan GKE untuk berjalan di kumpulan node yang dapat dihentikan untuk beban kerja singkat juga menjadi poin plus.

Membuktikan komitmen kami pada hardening keamanan

Namun, apabila ingin maju, kami perlu mendokumentasikan kebijakan dan proses pengelolaan keamanan informasi dan membuktikan bahwa kami mematuhi praktik terbaik untuk hardening keamanan.

Secara khusus, terkait sertifikasi ISO-27001, kami perlu mematuhi proses umumnya:

1. Mendokumentasikan proses yang dilakukan untuk mencapai kepatuhan
2. Membuktikan bahwa proses tersebut benar-benar merealisasikan tujuan kepatuhan
3. Memberikan bukti bahwa Anda mengikuti prosesnya
4. Mendokumentasikan setiap penyimpangan atau pengecualian

Meskipun Google Cloud menawarkan panduan hardening untuk GKE dan beberapa blog GCP memandu pendekatan kami, kami masih perlu membuktikan bahwa kami memiliki praktik terbaik keamanan untuk sistem penting kami. Dengan bermunculannya teknologi baru, membuktikan adanya praktik terbaik tersebut kepada auditor dapat sulit dilakukan. Alasannya, buktinya sering berbentuk postingan blog yang diposting oleh kontributor utama dan pimpinan komunitas, bukan praktik terbaik resmi yang terdokumentasi. Untungnya, standar untuk Kubernetes mulai muncul. Baru-baru ini, Center for Internet Security (CIS) memublikasikan benchmark kepatuhan baru untuk Kubernetes 1.11 yang cukup komprehensif. Anda bahkan dapat menjalankan pemeriksaan otomatis berdasarkan CIS Benchmark menggunakan project open source yang luar biasa, kube-bench. Pada akhirnya, fakta bahwa Google mengelola infrastruktur GKE dasar membantu mempercepat proses sertifikasi.

Kepatuhan tanpa repot berkat GKE

Seperti disebutkan di atas, salah satu alasan kami beralih dari Kubernetes lokal ke GKE adalah agar kami tidak perlu berfokus pada pemeliharaan dan upgrade cluster Kubernetes secara manual, termasuk inisiatif kepatuhan kami. Berkat GKE, beban kerja tim kami berkurang karena Google mengelola dan mendokumentasikan sebagian besar

infrastruktur dasar. Kini kami dapat berfokus pada penyempurnaan dan pendokumentasian bagian-bagian prosedur keamanan yang unik bagi perusahaan dan industri kami, tanpa perlu mendokumentasikan semua teknologi dasar di infrastruktur kami.

Untuk Kubernetes, berikut adalah cuplikan cara kami mendokumentasikan infrastruktur menggunakan empat langkah yang dijelaskan di atas:

1. Kami menerapkan praktik terbaik keamanan dalam cluster Kubernetes dengan memastikan semua cluster dibandingkan terhadap panduan CIS Kubernetes. Kami menggunakan kube-bench untuk proses ini, yang dijalankan di cluster setiap tiga bulan sekali.

2. Otoritas pihak ketiga yang kredibel menerbitkan benchmark ini, yang mengonfirmasi bahwa proses kami menerapkan praktik terbaik dalam menggunakan Kubernetes dengan aman.

3. Kami menyediakan dokumentasi bahwa kami telah menilai cluster Kubernetes berdasarkan benchmark tersebut, termasuk tiket pelacakan tugas.

4. Kami menyediakan hasil penilaian dan mendokumentasikan pengecualian kebijakan dan bukti bahwa kami sudah mengevaluasi pengecualian tersebut berdasarkan metodologi pengelolaan risiko kami.

Sama halnya dengan bagian keamanan fisik dalam standar ISO-27001, CIS Benchmark memiliki banyak bagian yang khusus membahas setelan keamanan untuk master dan node Kubernetes. Karena kami berjalan di GKE, Google menangani 95 dari 104 item baris di benchmark yang berlaku untuk infrastruktur kami. Untuk item yang tidak bisa dinilai berdasarkan benchmark (karena GKE tidak menampilkan masternya), kami menyediakan link ke dokumentasi keamanan Google terkait fitur tersebut (baca Kepercayaan Cluster dan Keamanan Bidang Kontrol). Berikut adalah beberapa contohnya:

- Menghubungkan kubelet ke master

- Menangani file config di master (misalnya scheduler, pengelola pengontrol, server API, dll.)
- Hardening database etcd

Selain GKE, kami juga berhasil memanfaatkan layanan Google Cloud lainnya yang memudahkan kami mengamankan jejak cloud (meski model tanggung jawab bersama untuk keamanan berarti Anda tidak dapat mengandalkan Google Cloud saja):

- Untuk praktik terbaik keamanan level OS, kami dapat mendokumentasikan praktik terbaik keamanan yang kuat untuk keamanan OS kami karena kami menggunakan Container-Optimized OS (COS) Google yang menyediakan banyak praktik terbaik keamanan secara default menggunakan hal-hal, seperti sistem file baca-saja. Yang perlu kami lakukan hanyalah mengikuti praktik terbaik demi membantu mengamankan beban kerja kami.
- Kami menggunakan auto-upgrade node di node GKE untuk menangani pengelolaan patch di lapisan OS untuk node kami. Untuk tingkat usahanya, kami mendapati bahwa auto-upgrade node memberikan keseimbangan yang baik antara patching dan stabilitas. Sampai hari ini, software kami belum pernah mengalami masalah sebagai akibat penggunaan auto-upgrade node.
- Kami menggunakan Container Analysis (yang merupakan fitur bawaan di Google Container Registry) untuk memindai kerentanan yang diketahui di image Docker kami.
- ISO-27001 mengharuskan Anda menunjukkan keamanan fisik dari infrastruktur jaringan Anda. Karena menjalankan seluruh infrastruktur di cloud, kami dapat langsung mengandalkan keamanan fisik dan jaringan Google Cloud untuk sebagian sertifikasi (Google Cloud sudah mendapatkan sertifikasi ISO-27001, selain sertifikasi lainnya).

DroneDeploy bertekad untuk memberikan akses pencitraan udara dan teknologi pemetaan yang mudah dan cepat kepada pelanggan kami. Kami menangani berbagai informasi sensitif atas nama pelanggan. Selain itu, kami ingin mereka tahu bahwa kami menerapkan praktik terbaik keamanan meski teknologi dasarnya menjadi makin kompleks, seperti kasus Kubernetes. Bagi DroneDeploy, peralihan ke GKE dan Google Cloud membantu kami menekan overhead operasional dan mempercepat langkah kami dalam memperoleh sertifikasi kepatuhan utama.

Referensi lebih lanjut

- cloud.google.com/containers/security
- “Anthos: An opportunity to modernize security”
- Dokumentasi keamanan Google Kubernetes Engine