aws

Panduan Developerr

Device Farm AWS

Versi API 2015-06-23

Device Farm AWS: Panduan Developerr

Table of Contents

Apa itu AWS Device Farm?

Device Farm adalah layanan pengujian aplikasi yang dapat Anda gunakan untuk menguji dan berinteraksi dengan aplikasi Android, iOS, dan web Anda di ponsel dan tablet fisik nyata yang di-host oleh Amazon Web Services (AWS).

Ada dua cara utama untuk menggunakan Device Farm:

- Pengujian otomatis aplikasi menggunakan berbagai kerangka pengujian.
- Akses jarak jauh perangkat tempat Anda dapat memuat, menjalankan, dan berinteraksi dengan aplikasi secara real time.

> **Note**
>
> Device Farm hanya tersedia di`us-west-2`Wilayah (Oregon).

Pengujian aplikasi otomatis

Device Farm memungkinkan Anda mengunggah pengujian Anda sendiri atau menggunakan uji kompatibilitas bawaan bebas skrip. Karena pengujian dilakukan secara paralel, pengujian pada beberapa perangkat dimulai dalam hitungan menit.

Saat tes selesai, laporan pengujian yang berisi hasil tingkat tinggi, log tingkat rendah,pixel-to-pixeltangkapan layar, dan data kinerja diperbarui.

Device Farm mendukung pengujian aplikasi Android dan iOS asli dan hybrid, termasuk yang dibuat denganPhoneGap, Titanium, Xamarin, Unity, dan kerangka kerja lainnya. Ini mendukung akses jarak jauh aplikasi Android dan iOS untuk pengujian interaktif. Untuk informasi selengkapnya tentang jenis pengujian yang didukung, lihatBekerja dengan jenis pengujian di AWS Device Farm.

Interaksi akses jarak jauh

Akses jarak jauh memungkinkan Anda untuk menggesek, memberi isyarat, dan berinteraksi dengan perangkat melalui browser web Anda secara real time. Ada sejumlah situasi di mana interaksi real-time dengan perangkat berguna. Misalnya, perwakilan layanan pelanggan dapat

memandu pelanggan melalui penggunaan atau pengaturan perangkat mereka. Mereka juga dapat memandu pelanggan melalui penggunaan aplikasi yang berjalan pada perangkat tertentu. Anda dapat menginstal aplikasi pada perangkat yang berjalan dalam sesi akses jarak jauh dan kemudian mereproduksi masalah pelanggan atau bug yang dilaporkan.

Selama sesi akses jarak jauh, Device Farm mengumpulkan detail tentang tindakan yang terjadi saat Anda berinteraksi dengan perangkat. Log dengan detail ini dan pengambilan video sesi diproduksi di akhir sesi.

Terminologi

Device Farm memperkenalkan istilah-istilah berikut yang menentukan cara informasi diatur:

perangkat kolam

Kumpulan perangkat yang biasanya memiliki karakteristik serupa, seperti platform, pabrikan, atau model.

pekerjaan

Permintaan Device Farm untuk menguji satu aplikasi terhadap satu perangkat. Sebuah pekerjaan berisi satu atau lebih suite.

pengukuran

Mengacu pada penagihan untuk perangkat. Anda mungkin melihat referensi ke perangkat terukur atau perangkat yang tidak diukur dalam dokumentasi dan referensi API. Untuk informasi selengkapnya tentang harga, lihatHarga AWS Device Farm.

proyek

Ruang kerja logis yang berisi run, satu run untuk setiap pengujian satu aplikasi terhadap satu perangkat atau beberapa. Anda dapat menggunakan proyek untuk mengatur ruang kerja dengan cara apa pun yang Anda pilih. Misalnya, Anda dapat memiliki satu proyek per judul aplikasi atau satu proyek per platform. Anda dapat membuat proyek sebanyak yang Anda butuhkan.

laporan

Berisi informasi tentang proses, yang merupakan permintaan Device Farm untuk menguji satu aplikasi terhadap satu atau beberapa perangkat. Untuk informasi selengkapnya, lihat Laporan di AWS Device Farm.

jalankan

Build spesifik aplikasi Anda, dengan serangkaian pengujian tertentu, akan dijalankan pada satu set perangkat tertentu. Lari menghasilkan laporan hasil. Lari berisi satu atau lebih pekerjaan. Untuk informasi selengkapnya, lihat Berjalan.

sesi

Interaksi real-time dengan perangkat fisik aktual melalui browser web Anda. Untuk informasi selengkapnya, lihat Sesi.

suite

Organisasi hierarkis tes dalam paket uji. Suite berisi satu atau lebih tes.

pengujian

Kasus uji individu dalam paket uji.

Untuk informasi selengkapnya tentang Device Farm, lihatKonsep.

Pengaturan

Untuk menggunakan Device Farm, lihatMenyiapkan.

Menyiapkan AWS Device Farm

Sebelum Anda menggunakan Device Farm untuk pertama kalinya, Anda harus menyelesaikan tugas-tugas berikut:

Topik

Langkah 1: Mendaftar AWS

Mendaftar untuk Amazon Web Services (AWS).

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Langkah 2: Buat atau gunakan pengguna IAM di akun Anda AWS

Kami menyarankan Anda untuk tidak menggunakan akun AWS root Anda untuk mengakses Device Farm. Sebagai gantinya, buat pengguna AWS Identity and Access Management (IAM) (atau gunakan yang sudah ada) di AWS akun Anda, lalu akses Device Farm dengan pengguna IAM tersebut.

Untuk informasi selengkapnya, lihat Membuat Pengguna IAM (AWS Management Console).

Langkah 3: Berikan izin kepada pengguna IAM untuk mengakses Device Farm

Berikan izin kepada pengguna IAM untuk mengakses Device Farm. Untuk melakukannya, buat kebijakan akses di IAM, lalu tetapkan kebijakan akses ke pengguna IAM, sebagai berikut.

> **Note**
>
> Akun AWS root atau pengguna IAM yang Anda gunakan untuk menyelesaikan langkah-langkah berikut harus memiliki izin untuk membuat kebijakan IAM berikut dan melampirkannya ke pengguna IAM. Untuk informasi selengkapnya, lihat Bekerja dengan Kebijakan.

1. Buat kebijakan dengan badan JSON berikut. Berikan judul deskriptif, seperti *`DeviceFarmAdmin`*.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "devicefarm:*"
         ],
         "Resource": [
           "*"
         ]
       }
     ]
   }
   ```

   Untuk informasi selengkapnya tentang membuat kebijakan IAM, lihat Membuat Kebijakan IAM di Panduan Pengguna IAM.
2. Lampirkan kebijakan IAM yang Anda buat ke pengguna baru Anda. Untuk informasi selengkapnya tentang melampirkan kebijakan IAM ke pengguna, lihat Menambahkan dan Menghapus Kebijakan IAM di Panduan Pengguna IAM.

Melampirkan kebijakan memberi pengguna IAM akses ke semua tindakan dan sumber daya Device Farm yang terkait dengan pengguna IAM tersebut. Untuk informasi tentang cara membatasi pengguna IAM pada serangkaian tindakan dan sumber daya Device Farm terbatas, lihat. Manajemen identitas dan akses di AWS Device Farm

Langkah selanjutnya

Anda sekarang siap untuk mulai menggunakan Device Farm. Lihat Memulai dengan Device Farm.

Memulai dengan Device Farm

Panduan ini menunjukkan cara menggunakan Device Farm untuk menguji aplikasi Android atau iOS asli. Anda menggunakan konsol Device Farm untuk membuat proyek, mengunggah file.apk atau .ipa, menjalankan rangkaian pengujian standar, dan kemudian melihat hasilnya.

> **Note**
>
> Device Farm hanya tersedia di`us-west-2`(Oregon)AWSWilayah.

Topik

Prasyarat

Sebelum Anda mulai, pastikan Anda telah menyelesaikan persyaratan berikut:

- Selesaikan langkah-langkah dalam Menyiapkan. Anda membutuhkanAWSakun danAWS Identity and Access Management(IAM) pengguna dengan izin untuk mengakses Device Farm.

- Untuk Android, Anda memerlukan file.apk (paket aplikasi Android). Untuk iOS, Anda memerlukan file.ipa (arsip aplikasi iOS). Anda mengunggah file ke Device Farm nanti dalam panduan ini.

  > **Note**
  >
  > Pastikan file.ipa Anda dibuat untuk perangkat iOS dan bukan untuk simulator.

- (Opsional) Anda memerlukan pengujian dari salah satu kerangka pengujian yang didukung Device Farm. Anda mengunggah paket pengujian ini ke Device Farm, lalu jalankan pengujian nanti dalam

panduan ini. Jika Anda tidak memiliki paket pengujian yang tersedia, Anda dapat menentukan dan menjalankan rangkaian pengujian bawaan standar. Untuk informasi selengkapnya, lihat Bekerja dengan jenis pengujian di AWS Device Farm.

Langkah 1: Masuk ke konsol

Anda dapat menggunakan konsol Device Farm untuk membuat dan mengelola proyek dan berjalan untuk pengujian. Anda belajar tentang proyek dan berjalan nanti dalam panduan ini.

- Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.

Langkah 2: Buat proyek

Untuk menguji aplikasi di Device Farm, Anda harus terlebih dahulu membuat proyek.

1. Di panel navigasi, pilihPengujian Perangkat Seluler, dan kemudian pilihProyek.
2. Di bawahProyek Pengujian Perangkat Seluler, pilihProyek baru.
3. Di bawahBuat proyek, masukkanNama Proyek(Sebagai contoh,**`MyDemoProject`**).
4. Pilih Create (Buat).

   Konsol membukaTes otomatishalaman proyek Anda yang baru dibuat.

Langkah 3: Buat dan mulai lari

Sekarang setelah Anda memiliki proyek, Anda dapat membuat dan kemudian mulai menjalankan. Untuk informasi selengkapnya, lihat Berjalan.

1. PadaTes otomatishalaman, pilihBuat run baru.
2. PadaPilih aplikasihalaman, di bawahAplikasi Seluler, pilihPilih File, lalu pilih file Android (.apk) atau iOS (.ipa) dari komputer Anda. Atau, seret file dari komputer Anda dan letakkan di konsol.
3. MasukkanJalankan nama, seperti**`my first test`**. Secara default, konsol Device Farm menggunakan nama file.
4. Pilih Selanjutnya.

5. PadaKonfigurasikanhalaman, di bawahSetup kerangka uji, pilih salah satu kerangka pengujian atau suite pengujian bawaan. Untuk informasi tentang setiap opsi, lihat Jenis dan kerangka kerja uji.
   - Jika Anda belum mengemas pengujian untuk Device Farm, pilihBawaan: Fuzzuntuk menjalankan rangkaian pengujian bawaan standar. Anda dapat menyimpan nilai default untukHitungan acara,Throttle acara, danBiji pengacak. Untuk informasi selengkapnya, lihat the section called “Bawaan: fuzz (Android dan iOS)”.
   - Jika Anda memiliki paket pengujian dari salah satu kerangka pengujian yang didukung, pilih kerangka pengujian yang sesuai, lalu unggah file yang berisi pengujian Anda.
6. Pilih Selanjutnya.
7. PadaPilih perangkathalaman, untukKolam perangkat, pilihPerangkat Teratas.
8. Pilih Selanjutnya.
9. PadaTentukan status perangkathalaman, lakukan salah satu hal berikut:
   - Untuk memberikan data tambahan untuk Device Farm untuk digunakan selama dijalankan, di bawahTambahkan data tambahan, unggah file.zip.
   - Untuk menginstal aplikasi lain untuk dijalankan, di bawahInstal aplikasi lain, unggah file.apk atau .ipa untuk aplikasi. Untuk mengubah urutan instalasi, seret dan lepas file.
   - Untuk mengaktifkan radio Wi-Fi, Bluetooth, GPS, atau NFC untuk dijalankan, di bawahAtur status radio, pilih kotak centang yang sesuai.

     > **Note**
     >
     > Menyetel status radio perangkat hanya tersedia untuk pengujian asli Android saat ini.

   - Untuk menguji perilaku spesifik lokasi selama dijalankan, di bawahLokasi perangkat, tentukan presetLintangdanBujurkoordinat.
   - Untuk mengatur bahasa dan wilayah perangkat untuk dijalankan, di bawahPerangkat lokal, pilih lokal.
   - Untuk mengatur profil jaringan untuk dijalankan, di bawahProfil jaringan, pilih profil yang dikuratori. Atau, pilihBuat profil jaringanuntuk membuat milik Anda sendiri.
10. Pilih Selanjutnya.
11. PadaTinjau dan mulai jalankanhalaman, pilihKonfirmasikan dan mulai jalankan.

Device Farm mulai dijalankan segera setelah perangkat tersedia, biasanya dalam beberapa menit. Untuk melihat status run, padaTes otomatishalaman proyek Anda, pilih nama run Anda. Satu halaman run, di bawahPerangkat, setiap perangkat dimulai dengan ikon yang tertunda
tabel perangkat, lalu beralih ke ikon yang sedang berjalan
tes dimulai. Saat setiap pengujian selesai, konsol menampilkan ikon hasil pengujian di sebelah nama perangkat. Ketika semua pengujian selesai, ikon tertunda di sebelah run berubah menjadi ikon hasil pengujian.

Langkah 4: Lihat hasil run

Untuk melihat hasil pengujian dari proses, diTes otomatishalaman proyek Anda, pilih nama run Anda. Halaman ringkasan menampilkan:

- Jumlah total tes, berdasarkan hasil.
- Daftar tes dengan peringatan atau kegagalan unik.
- Daftar perangkat dengan hasil tes untuk masing-masing.
- Setiap tangkapan layar yang diambil selama proses, dikelompokkan berdasarkan perangkat.
- Bagian untuk mengunduh hasil parsing.

Untuk informasi selengkapnya, lihat Bekerja dengan laporan pengujian di Device Farm.

Langkah selanjutnya

Untuk informasi selengkapnya tentang Device Farm, lihatKonsep.

Beli slot perangkat di Device Farm

Anda dapat menggunakan konsol Device Farm,AWS Command Line Interface(AWS CLI), atau Device Farm API untuk membeli slot perangkat.

Topik

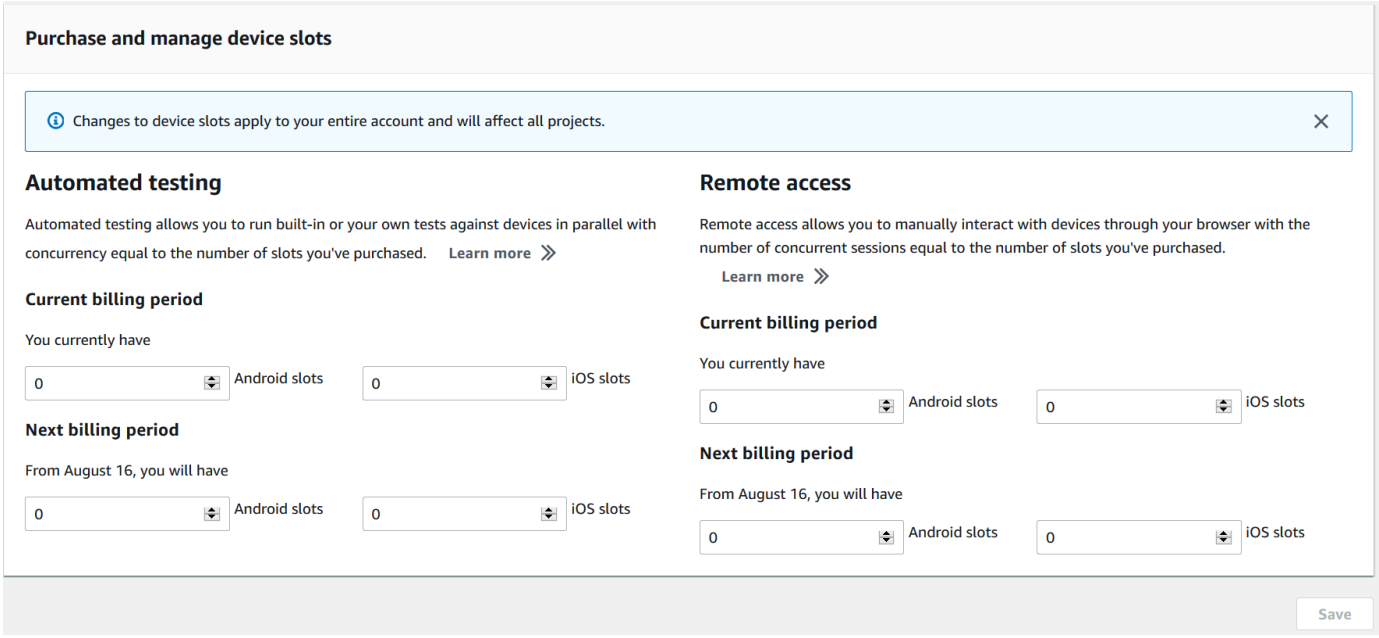
Beli slot perangkat (konsol)

1. Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.
2. Di panel navigasi, pilihPengujian Perangkat Seluler, dan kemudian pilihSlot perangkat.
3. PadaBeli dan kelola slot perangkathalaman, Anda dapat membuat paket kustom Anda sendiri dengan memilih jumlah slotPengujian otomatisdanAkses jarak jauhperangkat yang ingin Anda beli. Tentukan jumlah slot untuk periode penagihan saat ini dan berikutnya.

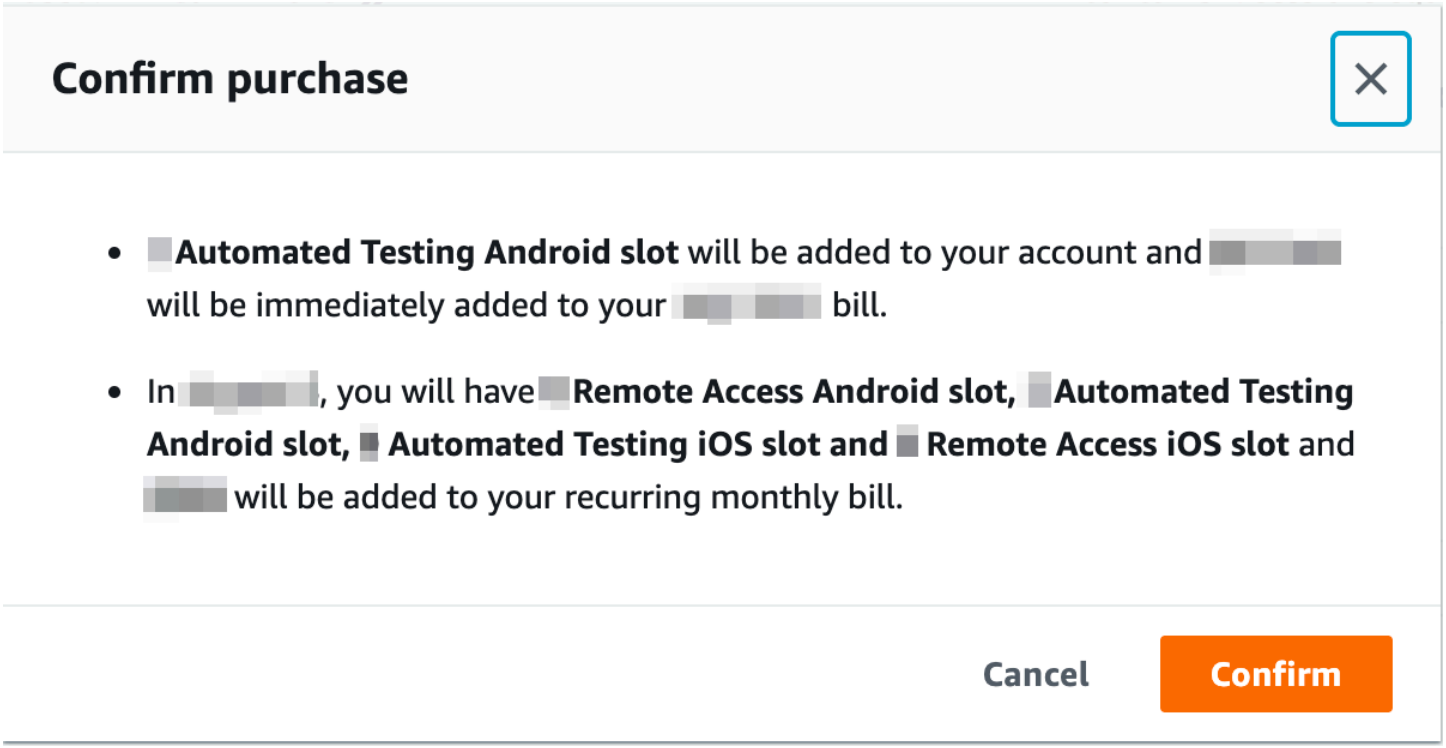
   Saat Anda mengubah jumlah slot, teks diperbarui secara dinamis dengan jumlah penagihan. Untuk informasi lebih lanjut, lihatHarga Pertanian Perangkat AWS.

   > **Important**
   >
   > Jika Anda mengubah jumlah slot perangkat tetapi lihathubungi kamiatauhubungi kami untuk membelipesan, AndaAWSakun belum disetujui untuk membeli jumlah slot perangkat yang Anda minta.
   > Opsi ini meminta Anda untuk mengirim email ke tim dukungan Device Farm. Di email, tentukan jumlah setiap jenis perangkat yang ingin Anda beli dan siklus penagihan mana.

   > **Note**
   >
   > Perubahan pada slot perangkat berlaku untuk seluruh akun Anda dan memengaruhi semua proyek.



4. Pilih Beli. SEBUAHKonfirmasikan pembelianjendela muncul. Tinjau informasinya, lalu pilihKonfirmasikanuntuk menyelesaikan transaksi.



PadaBeli dan kelola slot perangkathalaman, Anda dapat melihat jumlah slot perangkat yang saat ini Anda miliki. Jika Anda menambah atau mengurangi jumlah slot, Anda akan melihat jumlah slot yang akan Anda miliki satu bulan setelah tanggal Anda melakukan perubahan.

Beli slot perangkat (AWS CLI)

Anda dapat menjalankanpurchase-offeringperintah untuk membeli penawaran.

Untuk mencantumkan pengaturan akun Device Farm Anda, termasuk jumlah maksimum slot perangkat yang dapat Anda beli dan jumlah menit uji coba gratis yang tersisa, jalankanget-account-settingsperintah. Anda akan melihat output yang mirip dengan berikut ini:

```
{
    "accountSettings": {
        "maxSlots": {
            "GUID": 1,
            "GUID": 1,
            "GUID": 1,
            "GUID": 1
        },
        "unmeteredRemoteAccessDevices": {
            "ANDROID": 0,
            "IOS": 0
        },
        "maxJobTimeoutMinutes": 150,
        "trialMinutes": {
            "total": 1000.0,
            "remaining": 954.1
        },
        "defaultJobTimeoutMinutes": 150,
        "awsAccountNumber": "AWS-ACCOUNT-NUMBER",
        "unmeteredDevices": {
            "ANDROID": 0,
            "IOS": 0
        }
    }
}
```

Untuk membuat daftar penawaran slot perangkat yang tersedia untuk Anda, jalankanlist-offeringsperintah. Anda akan melihat output yang serupa dengan yang berikut:

```
{
```

```
    "offerings": [
        {
            "recurringCharges": [
                {
                    "cost": {
                        "amount": 250.0,
                        "currencyCode": "USD"
                    },
                    "frequency": "MONTHLY"
                }
            ],
            "platform": "IOS",
            "type": "RECURRING",
            "id": "GUID",
            "description": "iOS Unmetered Device Slot"
        },
        {
            "recurringCharges": [
                {
                    "cost": {
                        "amount": 250.0,
                        "currencyCode": "USD"
                    },
                    "frequency": "MONTHLY"
                }
            ],
            "platform": "ANDROID",
            "type": "RECURRING",
            "id": "GUID",
            "description": "Android Unmetered Device Slot"
        },
        {
            "recurringCharges": [
                {
                    "cost": {
                        "amount": 250.0,
                        "currencyCode": "USD"
                    },
                    "frequency": "MONTHLY"
                }
            ],
            "platform": "ANDROID",
            "type": "RECURRING",
            "id": "GUID",
```

```
            "description": "Android Remote Access Unmetered Device Slot"
        },
        {
            "recurringCharges": [
                {
                    "cost": {
                        "amount": 250.0,
                        "currencyCode": "USD"
                    },
                    "frequency": "MONTHLY"
                }
            ],
            "platform": "IOS",
            "type": "RECURRING",
            "id": "GUID",
            "description": "iOS Remote Access Unmetered Device Slot"
        }
    ]
}
```

Untuk membuat daftar promosi penawaran yang tersedia, jalankanlist-offering-promotionsperintah.

> **Note**
>
> Perintah ini hanya mengembalikan promosi yang belum Anda beli. Segera setelah Anda membeli satu atau lebih slot di penawaran apa pun menggunakan promosi, promosi itu tidak lagi muncul dalam hasil.

Anda akan melihat output yang serupa dengan yang berikut:

```
{
    "offeringPromotions": [
        {
            "id": "2FREEMONTHS",
            "description": "New device slot customers get 3 months for the price of 1."
        }
    ]
}
```

Untuk mendapatkan status penawaran, jalankanget-offering-statusperintah. Anda akan melihat output yang serupa dengan yang berikut:

```
{
    "current": {
        "GUID": {
            "offering": {
                "platform": "IOS",
                "type": "RECURRING",
                "id": "GUID",
                "description": "iOS Unmetered Device Slot"
            },
            "quantity": 1
        },
        "GUID": {
            "offering": {
                "platform": "ANDROID",
                "type": "RECURRING",
                "id": "GUID",
                "description": "Android Unmetered Device Slot"
            },
            "quantity": 1
        }
    },
    "nextPeriod": {
        "GUID": {
            "effectiveOn": 1459468800.0,
            "offering": {
                "platform": "IOS",
                "type": "RECURRING",
                "id": "GUID",
                "description": "iOS Unmetered Device Slot"
            },
            "quantity": 1
        },
        "GUID": {
            "effectiveOn": 1459468800.0,
            "offering": {
                "platform": "ANDROID",
                "type": "RECURRING",
                "id": "GUID",
                "description": "Android Unmetered Device Slot"
            },
            "quantity": 1
        }
    }
```

```
}
```

Therenew-offeringdanlist-offering-transactionsperintah juga tersedia untuk fitur ini. Untuk informasi lain, lihat Referensi AWS CLI.

Beli slot perangkat (API)

1. PanggilGetAccountSettingsoperasi untuk daftar pengaturan akun Anda.
2. PanggilListOfferingsoperasi untuk daftar penawaran slot perangkat yang tersedia untuk Anda.
3. PanggilListOfferingPromotionsoperasi untuk daftar promosi penawaran yang tersedia.

   > **Note**
   >
   > Perintah ini hanya mengembalikan promosi yang belum Anda beli. Segera setelah Anda membeli satu atau lebih slot menggunakan promosi penawaran, promosi itu tidak lagi muncul dalam hasil.

4. PanggilPurchaseOfferingoperasi untuk membeli penawaran.
5. PanggilGetOfferingStatusoperasi untuk mendapatkan status penawaran.

TheRenewOfferingdanListOfferingTransactionsperintah juga tersedia untuk fitur ini.

Untuk informasi tentang menggunakan Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Konsep AWS Device Farm

Bagian ini menjelaskan konsep-konsep penting Device Farm.

- Dukungan perangkat di AWS Device Farm
- Lingkungan uji
- Berjalan
- Laporan di AWS Device Farm
- Sesi

Untuk informasi selengkapnya tentang jenis pengujian yang didukung di Device Farm, lihatBekerja dengan jenis pengujian di AWS Device Farm.

Dukungan perangkat di AWS Device Farm

Bagian berikut memberikan informasi tentang dukungan perangkat di Device Farm.

Topik

- Perangkat yang didukung
- Kolam perangkat
- Perangkat pribadi
- Pencitraan merek perangkat
- Slot perangkat
- Aplikasi perangkat yang sudah diinstal sebelumnya
- Kemampuan perangkat

Perangkat yang didukung

Device Farm menyediakan dukungan untuk ratusan perangkat Android dan iOS yang unik dan populer serta kombinasi sistem operasi. Daftar perangkat yang tersedia bertambah saat perangkat baru memasuki pasar. Untuk daftar lengkap perangkat, lihat Daftar Perangkat.

Kolam perangkat

Device Farm mengatur perangkatnya ke dalam kumpulan perangkat yang dapat Anda gunakan untuk pengujian. Kumpulan perangkat ini berisi perangkat terkait, seperti perangkat yang hanya berjalan di Android atau hanya di iOS. Device Farm menyediakan kumpulan perangkat yang dikuratori, seperti untuk perangkat teratas. Anda juga dapat membuat kumpulan perangkat yang memadukan perangkat publik dan pribadi.

Perangkat pribadi

Perangkat pribadi memungkinkan Anda menentukan konfigurasi perangkat keras dan perangkat lunak yang tepat untuk kebutuhan pengujian Anda. Konfigurasi tertentu, seperti perangkat Android yang di-rooting, dapat didukung sebagai perangkat pribadi. Setiap perangkat pribadi adalah perangkat fisik yang digunakan Device Farm atas nama Anda di pusat data Amazon. Perangkat pribadi Anda tersedia secara eksklusif untuk Anda untuk pengujian otomatis dan manual. Setelah Anda memilih untuk mengakhiri langganan Anda, perangkat keras dihapus dari lingkungan kami. Untuk informasi selengkapnya, lihat Perangkat Pribadi danBekerja dengan perangkat pribadi di AWS Device Farm.

Pencitraan merek perangkat

Device Farm menjalankan pengujian pada perangkat seluler dan tablet fisik dari berbagai OEM.

Slot perangkat

Slot perangkat sesuai dengan konkurensi di mana jumlah slot perangkat yang telah Anda beli menentukan berapa banyak perangkat yang dapat Anda jalankan dalam pengujian atau sesi akses jarak jauh.

Ada dua jenis slot perangkat:

- Slot perangkat akses jarak jauh adalah slot yang dapat Anda jalankan dalam sesi akses jarak jauh secara bersamaan.

  Jika Anda memiliki satu slot perangkat akses jarak jauh, Anda hanya dapat menjalankan satu sesi akses jarak jauh pada satu waktu. Jika Anda membeli slot perangkat pengujian jarak jauh tambahan, Anda dapat menjalankan beberapa sesi secara bersamaan.
- Slot perangkat pengujian otomatis adalah slot di mana Anda dapat menjalankan pengujian secara bersamaan.

Jika Anda memiliki satu slot perangkat pengujian otomatis, Anda hanya dapat menjalankan tes pada satu perangkat pada satu waktu. Jika Anda membeli slot perangkat pengujian otomatis tambahan, Anda dapat menjalankan beberapa pengujian secara bersamaan, di beberapa perangkat, untuk mendapatkan hasil pengujian lebih cepat.

Anda dapat membeli slot perangkat berdasarkan keluarga perangkat (perangkat Android atau iOS untuk pengujian otomatis dan perangkat Android atau iOS untuk akses jarak jauh). Untuk informasi selengkapnya, lihat Harga Device Farm.

Aplikasi perangkat yang sudah diinstal sebelumnya

Perangkat di Device Farm menyertakan sejumlah kecil aplikasi yang sudah diinstal oleh produsen dan operator.

Kemampuan perangkat

Semua perangkat memiliki koneksi Wi-Fi ke internet. Mereka tidak memiliki koneksi operator dan tidak dapat melakukan panggilan telepon atau mengirim pesan SMS.

Anda dapat mengambil foto dengan perangkat apa pun yang mendukung kamera depan atau belakang. Karena cara perangkat dipasang, foto mungkin terlihat gelap dan buram.

Layanan Google Play diinstal pada perangkat yang mendukungnya, tetapi perangkat ini tidak memiliki akun Google yang aktif.

Menguji lingkungan di AWS Device Farm

AWS Device Farm menyediakan lingkungan pengujian khusus dan standar untuk menjalankan pengujian otomatis Anda. Anda dapat memilih lingkungan pengujian khusus untuk kontrol penuh atas pengujian otomatis Anda. Atau, Anda dapat memilih lingkungan pengujian standar default Device Farm, yang menawarkan pelaporan terperinci dari setiap pengujian dalam rangkaian pengujian otomatis Anda.

Topik

Lingkungan uji standar

Saat Anda menjalankan pengujian di lingkungan standar, Device Farm menyediakan log dan pelaporan terperinci untuk setiap kasus dalam rangkaian pengujian Anda. Anda dapat melihat data kinerja, video, tangkapan layar, dan log untuk setiap pengujian untuk menentukan dan memperbaiki masalah di aplikasi Anda.

> **Note**
>
> Karena Device Farm menyediakan pelaporan terperinci di lingkungan standar, waktu eksekusi pengujian bisa lebih lama daripada saat Anda menjalankan pengujian secara lokal. Jika Anda ingin waktu eksekusi lebih cepat, jalankan pengujian Anda di lingkungan pengujian khusus.

Lingkungan uji kustom

Saat menyesuaikan lingkungan pengujian, Anda dapat menentukan perintah yang harus dijalankan Device Farm untuk menjalankan pengujian. Ini memastikan bahwa pengujian di Device Farm berjalan dengan cara yang mirip dengan pengujian yang dijalankan di mesin lokal Anda. Menjalankan pengujian Anda dalam mode ini juga memungkinkan live log dan streaming video pengujian Anda. Saat Anda menjalankan pengujian di lingkungan pengujian yang disesuaikan, Anda tidak mendapatkan laporan terperinci untuk setiap kasus uji. Untuk informasi selengkapnya, lihat Bekerja dengan lingkungan pengujian khusus.

Anda memiliki opsi untuk menggunakan lingkungan pengujian khusus saat menggunakan konsol Device FarmAWS CLI, atau Device Farm API untuk membuat uji coba.

Untuk informasi selengkapnya, lihat Mengunggah Spesifikasi Uji Kustom Menggunakan dan. AWS CLI Membuat uji coba di Device Farm

Berjalan di AWS Device Farm

Bagian berikut berisi informasi tentang berjalan di Device Farm.

Jalankan di Device Farm mewakili build spesifik aplikasi Anda, dengan serangkaian pengujian tertentu, untuk dijalankan pada set perangkat tertentu. Run menghasilkan laporan yang berisi informasi tentang hasil run. Lari berisi satu atau lebih pekerjaan.

Topik

Jalankan konfigurasi

Sebagai bagian dari proses, Anda dapat menyediakan pengaturan yang dapat digunakan Device Farm untuk mengganti pengaturan perangkat saat ini. Ini termasuk koordinat lintang dan bujur, lokal, status radio (seperti Bluetooth, GPS, NFC, dan Wi-Fi), data tambahan (terkandung dalam file.zip), dan aplikasi tambahan (aplikasi yang harus diinstal sebelum aplikasi akan diuji).

Jalankan retensi file

Device Farm menyimpan aplikasi dan file Anda selama 30 hari dan kemudian menghapusnya dari sistemnya. Namun, Anda dapat menghapus file Anda kapan saja.

Device Farm menyimpan hasil run, log, dan screenshot Anda selama 400 hari dan kemudian menghapusnya dari sistemnya.

Jalankan status perangkat

Device Farm selalu me-reboot perangkat sebelum membuatnya tersedia untuk pekerjaan berikutnya.

Berjalan paralel

Device Farm menjalankan pengujian secara paralel saat perangkat tersedia.

Mengatur batas waktu eksekusi

Anda dapat menetapkan nilai berapa lama uji coba harus dijalankan sebelum menghentikan setiap perangkat menjalankan pengujian. Misalnya, jika pengujian Anda membutuhkan waktu 20 menit per perangkat untuk diselesaikan, Anda harus memilih batas waktu 30 menit per perangkat.

Untuk informasi selengkapnya, lihat Tetapkan batas waktu eksekusi untuk pengujian yang dijalankan di AWS Device Farm.

Aplikasi instrumentasi

Anda tidak perlu menginstruksikan aplikasi Anda atau menyediakan Device Farm dengan kode sumber untuk aplikasi Anda. Aplikasi Android dapat dikirimkan tanpa dimodifikasi. Aplikasi iOS harus dibangun dengan target Perangkat iOS, bukan dengan simulator.

Menandatangani ulang aplikasi dalam proses

Untuk aplikasi iOS, Anda tidak perlu menambahkan UUID Device Farm apa pun ke profil penyediaan Anda. Device Farm mengganti profil penyediaan yang disematkan dengan profil wildcard dan kemudian menandatangani ulang aplikasi. Jika Anda memberikan data tambahan, Device Farm menambahkannya ke paket aplikasi sebelum Device Farm menginstalnya, sehingga tambahan tersebut ada di kotak pasir aplikasi Anda. Menandatangani ulang aplikasi menghapus hak seperti Grup Aplikasi, Domain Terkait, Game Center,,, Konfigurasi Aksesori Nirkabel HealthKit HomeKit, Pembelian Dalam Aplikasi, Audio Antar-Aplikasi, Apple Pay, Pemberitahuan Push, dan Konfigurasi & Kontrol VPN.

Untuk aplikasi Android, Device Farm menandatangani ulang aplikasi. Ini dapat merusak fungsionalitas apa pun yang bergantung pada tanda tangan aplikasi, seperti Google Maps Android API, atau mungkin memicu antipiracy atau deteksi antitamper dari produk seperti. DexGuard

Aplikasi yang dikaburkan sedang berjalan

Untuk aplikasi Android, jika aplikasi dikaburkan, Anda masih dapat mengujinya dengan Device Farm jika Anda menggunakannya. ProGuard Namun, jika Anda menggunakan DexGuard dengan tindakan antipembajakan, Device Farm tidak dapat menandatangani ulang dan menjalankan pengujian terhadap aplikasi.

Iklan berjalan

Sebaiknya hapus iklan dari aplikasi sebelum mengunggahnya ke Device Farm. Kami tidak dapat menjamin bahwa iklan ditampilkan selama berjalan.

Media sedang berjalan

Anda dapat menyediakan media atau data lain untuk menemani aplikasi Anda. Data tambahan harus disediakan dalam file.zip berukuran tidak lebih dari 4 GB.

Tugas umum untuk berjalan

Lihat informasi yang lebih lengkap di Membuat uji coba di Device Farm dan Bekerja dengan uji coba di AWS Device Farm.

Laporan di AWS Device Farm

Bagian berikut memberikan informasi tentang laporan pengujian Device Farm.

Topik

- Laporkan retensi
- Laporkan komponen
- Log dalam laporan
- Tugas umum untuk laporan

Laporkan retensi

Device Farm menyimpan laporan Anda selama 400 hari. Laporan ini mencakup metadata, log, tangkapan layar, dan data kinerja.

Laporkan komponen

Laporan di Device Farm berisi informasi lulus dan gagal, laporan kerusakan, log pengujian dan perangkat, tangkapan layar, dan data kinerja.

Laporan mencakup data per-perangkat rinci dan hasil tingkat tinggi, seperti jumlah kemunculan masalah yang diberikan.

Log dalam laporan

Laporan mencakup tangkapan logcat lengkap untuk pengujian Android dan log Konsol Perangkat lengkap untuk pengujian iOS.

Tugas umum untuk laporan

Lihat informasi yang lebih lengkap di Bekerja dengan laporan pengujian di Device Farm.

Sesi di AWS Device Farm

Anda dapat menggunakan Device Farm untuk melakukan pengujian interaktif aplikasi Android dan iOS melalui sesi akses jarak jauh di browser web. Pengujian interaktif semacam ini membantu teknisi dukungan pada panggilan pelanggan untuk melewati, langkah demi langkah, masalah pelanggan. Pengembang dapat mereproduksi masalah pada perangkat tertentu untuk mengisolasi kemungkinan sumber masalah. Anda dapat menggunakan sesi jarak jauh untuk melakukan tes kegunaan dengan pelanggan target Anda.

Topik

Perangkat yang didukung untuk akses jarak jauh

Device Farm menyediakan dukungan untuk sejumlah perangkat Android dan iOS yang unik dan populer. Daftar perangkat yang tersedia bertambah saat perangkat baru memasuki pasar. Konsol Device Farm menampilkan daftar perangkat Android dan iOS saat ini yang tersedia untuk akses jarak jauh. Untuk informasi selengkapnya, lihat Dukungan perangkat di AWS Device Farm.

Retensi file sesi

Device Farm menyimpan aplikasi dan file Anda selama 30 hari dan kemudian menghapusnya dari sistemnya. Namun, Anda dapat menghapus file Anda kapan saja.

Device Farm menyimpan log sesi Anda dan merekam video selama 400 hari dan kemudian menghapusnya dari sistemnya.

Aplikasi instrumentasi

Anda tidak perlu menginstruksikan aplikasi Anda atau menyediakan kode sumber untuk aplikasi Anda kepada Device Farm. Aplikasi Android dan iOS dapat dikirimkan tanpa dimodifikasi.

Menandatangani ulang aplikasi dalam sesi

Device Farm menandatangani ulang aplikasi Android dan iOS. Ini dapat merusak fungsionalitas yang bergantung pada tanda tangan aplikasi. Misalnya, Google Maps API untuk Android bergantung pada tanda tangan aplikasi Anda. Penandatanganan ulang aplikasi juga dapat memicu deteksi antipembajakan atau antitamper dari produk sepertiDexGuarduntuk perangkat Android.

Aplikasi yang dikaburkan dalam sesi

Untuk aplikasi Android, jika aplikasi dikaburkan, Anda masih dapat mengujinya dengan Device Farm jika Anda menggunakanProGuard. Namun, jika Anda menggunakanDexGuarddengan tindakan antipembajakan, Device Farm tidak dapat menandatangani ulang aplikasi.

Bekerja dengan proyek di AWS Device Farm

Project di Device Farm mewakili ruang kerja logis di Device Farm yang berisi run, satu run untuk setiap pengujian satu aplikasi terhadap satu perangkat atau beberapa perangkat. Proyek memungkinkan Anda untuk mengatur ruang kerja dengan cara apa pun yang Anda pilih. Misalnya, mungkin ada satu proyek per judul aplikasi, atau mungkin ada satu proyek per platform. Anda dapat membuat proyek sebanyak yang Anda butuhkan.

Anda dapat menggunakan konsol AWS Device Farm,AWS Command Line Interface(AWS CLI), atau AWS Device Farm API untuk bekerja dengan proyek.

Topik

Membuat proyek di AWS Device Farm

Anda dapat membuat proyek dengan menggunakan konsol AWS Device Farm,AWS CLI, atau AWS Device Farm API.

Prasyarat

- Selesaikan langkah-langkah dalam Menyiapkan.

Buat proyek (konsol)

1. Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.
3. PilihProyek baru.
4. Masukkan nama untuk proyek Anda, lalu pilihKirim.
5. Untuk menentukan pengaturan proyek, pilihPengaturan proyek. Pengaturan ini mencakup batas waktu default untuk uji coba. Setelah pengaturan diterapkan, mereka digunakan oleh semua uji coba untuk proyek. Untuk informasi selengkapnya, lihat Tetapkan batas waktu eksekusi untuk pengujian yang dijalankan di AWS Device Farm.

Buat proyek (AWS CLI)

- Jalankancreate-project, menentukan nama proyek.

  Contoh:

  ```
  aws devicefarm create-project --name MyProjectName
  ```

  TheAWS CLIrespon termasuk Amazon Resource Name (ARN) proyek.

  ```
  {
      "project": {
          "name": "MyProjectName",
          "arn": "arn:aws:devicefarm:us-west-2:123456789101:project:5e01a8c7-
  c861-4c0a-b1d5-12345EXAMPLE",
          "created": 1535675814.414
      }
  }
  ```

  Untuk informasi selengkapnya, lihat create-project dan Referensi AWS CLI.

Buat proyek (API)

- PanggilCreateProjectAPI.

Untuk informasi tentang menggunakan Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Lihat daftar proyek di AWS Device Farm

Anda dapat menggunakan konsol AWS Device Farm,AWS CLI, atau AWS Device Farm API untuk melihat daftar proyek.

Topik

Prasyarat

- Buat setidaknya satu proyek di Device Farm. Ikuti instruksi diMembuat proyek di AWS Device Farm, dan kemudian kembali ke halaman ini.

Lihat daftar proyek (konsol)

1. Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.
2. Untuk menemukan daftar proyek yang tersedia, lakukan hal berikut:
   - Untuk proyek pengujian perangkat seluler, pada menu navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.
   - Untuk proyek pengujian browser desktop, pada menu navigasi Device Farm, pilihPengujian Browser Desktop, lalu pilihProyek.

Lihat daftar proyek (AWS CLI)

- Untuk melihat daftar proyek, jalankanlist-projectsperintah.

  Untuk melihat informasi tentang satu proyek, jalankanget-projectperintah.

Untuk informasi tentang menggunakan Device Farm denganAWS CLI, lihatReferensi AWS CLI.

Lihat daftar proyek (API)

- Untuk melihat daftar proyek, hubungi`ListProjects`API.

  Untuk melihat informasi tentang satu proyek, hubungi`GetProject`API.

Untuk informasi tentang AWS Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Bekerja dengan uji coba di AWS Device Farm

Jalankan di Device Farm mewakili build spesifik aplikasi Anda, dengan serangkaian pengujian tertentu, untuk dijalankan pada set perangkat tertentu. Run menghasilkan laporan yang berisi informasi tentang hasil run. Lari berisi satu atau lebih pekerjaan. Untuk informasi selengkapnya, lihat Berjalan.

Anda dapat menggunakan konsol AWS Device Farm, AWS Command Line Interface (AWS CLI), atau AWS Device Farm API untuk bekerja dengan menjalankan.

Topik

Membuat uji coba di Device Farm

Anda dapat menggunakan konsol Device Farm AWS CLI, atau Device Farm API untuk membuat uji coba. Anda juga dapat menggunakan plugin yang didukung, seperti plugin Jenkins atau Gradle untuk Device Farm. Untuk informasi selengkapnya tentang plugin, lihat Alat dan plugin. Untuk informasi tentang lari, lihatBerjalan.

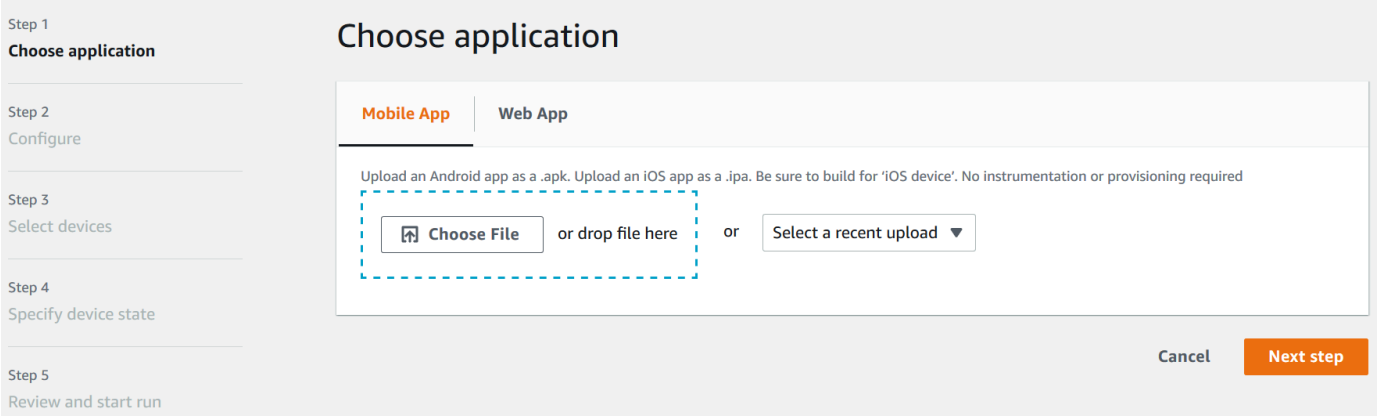
Topik

Prasyarat

Anda harus memiliki proyek di Device Farm. Ikuti instruksi diMembuat proyek di AWS Device Farm, dan kemudian kembali ke halaman ini.

Buat uji coba (konsol)

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Di panel navigasi, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Jika Anda sudah memiliki proyek, Anda dapat mengunggah tes Anda ke sana. Jika tidak, pilih Proyek baru, masukkan Nama Proyek, lalu pilih Buat.
4. Buka project Anda, lalu pilih Create a new run.
5. Pada halaman Pilih aplikasi, pilih Aplikasi Seluler atau Aplikasi Web.



6. Unggah file aplikasi Anda. Anda juga dapat menarik dan melepas file Anda atau memilih unggahan terbaru. Jika Anda mengunggah aplikasi iOS, pastikan untuk memilih perangkat iOS, bukan simulator.
7. (Opsional) Dalam nama Jalankan, masukkan nama. Secara default, Device Farm menggunakan nama file aplikasi.
8. Pilih Berikutnya.
9. Pada halaman Konfigurasi, pilih salah satu rangkaian pengujian yang tersedia.

> **Note**
>
> Jika Anda tidak memiliki pengujian yang tersedia, pilih Built-in: Fuzz untuk menjalankan rangkaian pengujian bawaan standar. Jika Anda memilih Built-in: Fuzz, dan kotak

benih Event count, Event throttle, dan Randomizer muncul, Anda dapat mengubah atau menyimpan nilainya.

Untuk informasi tentang rangkaian pengujian yang tersedia, lihatBekerja dengan jenis pengujian di AWS Device Farm.

10. Jika Anda tidak memilih Built-in: Fuzz, pilih Pilih File, lalu telusuri ke dan pilih file yang berisi pengujian Anda.
11. Untuk lingkungan pengujian Anda, pilih Jalankan pengujian Anda di lingkungan standar kami atau Jalankan pengujian Anda di lingkungan khusus. Untuk informasi selengkapnya, lihat Lingkungan uji.
12. Jika Anda menggunakan lingkungan pengujian standar, lewati ke langkah 13. Jika Anda menggunakan lingkungan pengujian khusus dengan file YAMM spesifikasi pengujian default, lewati ke langkah 13.

    a. Jika Anda ingin mengedit spesifikasi pengujian default di lingkungan pengujian kustom, pilih Edit untuk memperbarui spesifikasi YAMAL default.

    b. Jika Anda mengubah spesifikasi pengujian, pilih Simpan sebagai Baru untuk memperbaruinya.

13. Jika Anda ingin mengonfigurasi perekaman video atau opsi pengambilan data kinerja, pilih Konfigurasi Lanjutan.

    a. Pilih Aktifkan perekaman video untuk merekam video selama pengujian.

    b. Pilih Aktifkan pengambilan data kinerja aplikasi untuk menangkap data kinerja dari perangkat.

> **Note**
>
> Jika Anda memiliki perangkat pribadi, Konfigurasi khusus untuk Perangkat Pribadi juga ditampilkan.

14. Pilih Berikutnya.
15. Pada halaman Pilih perangkat, lakukan salah satu hal berikut:

    - Untuk memilih kumpulan perangkat bawaan untuk menjalankan pengujian, untuk kumpulan Perangkat, pilih Perangkat Teratas.

- Untuk membuat kumpulan perangkat Anda sendiri untuk menjalankan pengujian, ikuti petunjuk diBuat kumpulan perangkat, lalu kembali ke halaman ini.
- Jika Anda membuat kumpulan perangkat sendiri sebelumnya, untuk kumpulan Perangkat, pilih kumpulan perangkat Anda.

Untuk informasi selengkapnya, lihat Dukungan perangkat di AWS Device Farm.

16. Pilih Berikutnya.
17. Pada halaman Tentukan status perangkat:

    - Untuk menyediakan data lain bagi Device Farm untuk digunakan selama proses, di samping Tambahkan data tambahan, pilih Pilih File, lalu telusuri ke dan pilih file.zip yang berisi data.
    - Untuk menginstal aplikasi tambahan untuk Device Farm untuk digunakan selama menjalankan, di samping Instal aplikasi lain, pilih Pilih File, lalu telusuri ke dan pilih file.apk atau.ipa yang berisi aplikasi. Ulangi ini untuk aplikasi lain yang ingin Anda instal. Anda dapat mengubah urutan instalasi dengan menyeret dan menjatuhkan aplikasi setelah Anda mengunggahnya.
    - Untuk menentukan apakah Wi-Fi, Bluetooth, GPS, atau NFC diaktifkan selama proses, di samping Setel status radio, pilih kotak yang sesuai.
    - Untuk mengatur lintang dan bujur perangkat untuk menjalankan, di samping Lokasi perangkat, masukkan koordinat.
    - Untuk mengatur lokal perangkat untuk dijalankan, di lokal Perangkat, pilih lokal.
18. Pilih Berikutnya.
19. Pada halaman Tinjau dan mulai jalankan, Anda dapat menentukan batas waktu eksekusi untuk uji coba Anda. Jika Anda menggunakan slot pengujian tanpa batas, konfirmasikan bahwa Jalankan pada slot yang tidak diukur dipilih.
20. Masukkan nilai atau gunakan bilah geser untuk mengubah batas waktu eksekusi. Untuk informasi selengkapnya, lihat Tetapkan batas waktu eksekusi untuk pengujian yang dijalankan di AWS Device Farm.
21. Pilih Konfirmasi dan mulai jalankan.

Device Farm mulai dijalankan segera setelah perangkat tersedia, biasanya dalam beberapa menit. Selama uji coba, konsol Device Farm menampilkan ikon yang tertunda

di tabel run. Setiap perangkat yang sedang dijalankan juga akan dimulai dengan ikon yang tertunda, lalu beralih ke ikon yang sedang berjalan

saat pengujian dimulai. Saat setiap pengujian selesai, ikon hasil pengujian ditampilkan di sebelah nama perangkat. Ketika semua pengujian telah selesai, ikon yang tertunda di sebelah run berubah menjadi ikon hasil pengujian.

Jika Anda ingin menghentikan uji coba, lihatHentikan proses di AWS Device Farm.

Buat test run (AWS CLI)

Anda dapat menggunakan AWS CLI untuk membuat uji coba.

Topik

- Langkah 1: Pilih proyek
- Langkah 2: Pilih kumpulan perangkat
- Langkah 3: Unggah file aplikasi Anda
- Langkah 4: Unggah paket skrip pengujian Anda
- Langkah 5: (Opsional) Unggah spesifikasi pengujian khusus Anda
- Langkah 6: Jadwalkan uji coba

Langkah 1: Pilih proyek

Anda harus mengaitkan uji coba Anda dengan proyek Device Farm.

1. Untuk membuat daftar proyek Device Farm Anda, jalankanlist-projects. Jika Anda tidak memiliki proyek, lihatMembuat proyek di AWS Device Farm.

   Contoh:

   ```
   aws devicefarm list-projects
   ```

   Responsnya mencakup daftar proyek Device Farm Anda.

   ```
   {
       "projects": [
           {
   ```

```
            "name": "MyProject",
            "arn": "arn:aws:devicefarm:us-west-2:123456789101:project:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE",
            "created": 1503612890.057
        }
    ]
}
```

2. Pilih proyek yang akan dikaitkan dengan uji coba Anda, dan catat Nama Sumber Daya Amazon (ARN).

Langkah 2: Pilih kumpulan perangkat

Anda harus memilih kumpulan perangkat untuk dikaitkan dengan uji coba Anda.

1. Untuk melihat kumpulan perangkat Anda, jalankanlist-device-pools, tentukan ARN proyek Anda.

   Contoh:

   ```
   aws devicefarm list-device-pools --arn arn:MyProjectARN
   ```

   Responsnya mencakup kumpulan perangkat Device Farm bawaan, sepertiTop Devices, dan kumpulan perangkat apa pun yang sebelumnya dibuat untuk proyek ini:

   ```
   {
       "devicePools": [
           {
               "rules": [
                   {
                       "attribute": "ARN",
                       "operator": "IN",
                       "value": "[\"arn:aws:devicefarm:us-west-2::device:example1\",
   \"arn:aws:devicefarm:us-west-2::device:example2\",\"arn:aws:devicefarm:us-
   west-2::device:example3\"]"
                   }
               ],
               "type": "CURATED",
               "name": "Top Devices",
               "arn": "arn:aws:devicefarm:us-west-2::devicepool:example",
               "description": "Top devices"
           },
           {
   ```

```
            "rules": [
                {
                    "attribute": "PLATFORM",
                    "operator": "EQUALS",
                    "value": "\"ANDROID\""
                }
            ],
            "type": "PRIVATE",
            "name": "MyAndroidDevices",
            "arn": "arn:aws:devicefarm:us-west-2:605403973111:devicepool:example2"
        }
    ]
}
```

2. Pilih kumpulan perangkat, dan catat ARN-nya.

   Anda juga dapat membuat kumpulan perangkat, dan kemudian kembali ke langkah ini. Untuk informasi selengkapnya, lihat Buat kumpulan perangkat (AWS CLI).

Langkah 3: Unggah file aplikasi Anda

Untuk membuat permintaan unggahan dan mendapatkan URL unggahan yang telah ditetapkan sebelumnya dari Amazon Simple Storage Service (Amazon S3), Anda memerlukan:

- Proyek Anda ARN.
- Nama file aplikasi Anda.
- Jenis unggahan.

Untuk informasi selengkapnya, lihat create-upload.

1. Untuk mengunggah file, jalankan create-upload dengan`--project-arn`,`--name`, dan `--type` parameter.

   Contoh ini membuat unggahan untuk aplikasi Android:

   ```
   aws devicefarm create-upload --project-arn arn:MyProjectArn --name MyAndroid.apk --
   type ANDROID_APP
   ```

   Responsnya mencakup ARN unggahan aplikasi Anda dan URL yang telah ditetapkan sebelumnya.

```
{
    "upload": {
        "status": "INITIALIZED",
        "name": "MyAndroid.apk",
        "created": 1535732625.964,
        "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL",
        "type": "ANDROID_APP",
        "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE"
    }
}
```

2. Catat ARN unggahan aplikasi dan URL yang telah ditentukan sebelumnya.
3. Unggah file aplikasi Anda menggunakan URL presigned Amazon S3. Contoh ini digunakan curl untuk mengunggah file Android .apk:

```
curl -T MyAndroid.apk "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL"
```

Untuk informasi selengkapnya, lihat Mengunggah objek menggunakan URL yang telah ditetapkan sebelumnya di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

4. Untuk memeriksa status upload aplikasi Anda, jalankan get-upload dan tentukan ARN upload aplikasi.

```
aws devicefarm get-upload --arn arn:MyAppUploadARN
```

Tunggu hingga status dalam respons SUCCEEDED sebelum Anda mengunggah paket skrip pengujian Anda.

```
{
    "upload": {
        "status": "SUCCEEDED",
        "name": "MyAndroid.apk",
        "created": 1535732625.964,
        "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL",
        "type": "ANDROID_APP",
```

```
        "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE",
        "metadata": "{"valid": true}"
    }
}
```

Langkah 4: Unggah paket skrip pengujian Anda

Selanjutnya, Anda mengunggah paket skrip pengujian Anda.

1. Untuk membuat permintaan unggahan dan mendapatkan URL unggahan Amazon S3 yang telah ditetapkan sebelumnya, jalankan create-upload dengan parameter `--project-arn--name`, dan parameter. `--type`

   Contoh ini membuat upload paket uji Appium Java TestNG:

   ```
   aws devicefarm create-upload --project-arn arn:MyProjectARN --name MyTests.zip --
   type APPIUM_JAVA_TESTNG_TEST_PACKAGE
   ```

   Respons termasuk paket pengujian Anda mengunggah ARN dan URL yang telah ditetapkan sebelumnya.

   ```
   {
       "upload": {
           "status": "INITIALIZED",
           "name": "MyTests.zip",
           "created": 1535738627.195,
           "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
   ExampleURL",
           "type": "APPIUM_JAVA_TESTNG_TEST_PACKAGE",
           "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
   c861-4c0a-b1d5-12345EXAMPLE"
       }
   }
   ```

2. Catat ARN dari unggahan paket uji dan URL yang telah ditentukan sebelumnya.
3. Unggah file paket skrip pengujian Anda menggunakan URL presigned Amazon S3. Contoh ini digunakan curl untuk mengunggah file skrip Appium TestNG yang di-zip:

```
curl -T MyTests.zip "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL"
```

4. Untuk memeriksa status unggahan paket skrip pengujian Anda, jalankan get-upload dan tentukan ARN dari unggahan paket pengujian dari langkah 1.

```
aws devicefarm get-upload –-arn arn:MyTestsUploadARN
```

Tunggu hingga status dalam respons SUCCEEDED sebelum Anda melanjutkan ke langkah opsional berikutnya.

```
{
    "upload": {
        "status": "SUCCEEDED",
        "name": "MyTests.zip",
        "created": 1535738627.195,
        "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL",
        "type": "APPIUM_JAVA_TESTNG_TEST_PACKAGE",
        "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE",
        "metadata": "{"valid": true}"
    }
}
```

Langkah 5: (Opsional) Unggah spesifikasi pengujian khusus Anda

Jika Anda menjalankan pengujian di lingkungan pengujian standar, lewati langkah ini.

Device Farm mempertahankan file spesifikasi pengujian default untuk setiap jenis pengujian yang didukung. Selanjutnya, Anda mengunduh spesifikasi pengujian default dan menggunakannya untuk membuat unggahan spesifikasi pengujian khusus untuk menjalankan pengujian Anda di lingkungan pengujian khusus. Untuk informasi selengkapnya, lihat Lingkungan uji.

1. Untuk menemukan ARN upload untuk spesifikasi pengujian default Anda, jalankan list-uploads dan tentukan ARN proyek Anda.

```
aws devicefarm list-uploads --arn arn:MyProjectARN
```

Respons berisi entri untuk setiap spesifikasi pengujian default:

```
{
    "uploads": [
        {

            {
                "status": "SUCCEEDED",
                "name": "Default TestSpec for Android Appium Java TestNG",
                "created": 1529498177.474,
                "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL",
                "type": "APPIUM_JAVA_TESTNG_TEST_SPEC",
                "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE"
            }
        }
    ]
}
```

2. Pilih spesifikasi pengujian default Anda dari daftar. Catat unggahannya ARN.
3. Untuk mengunduh spesifikasi pengujian default Anda, jalankan get-upload dan tentukan ARN unggahan.

   Contoh:

   ```
   aws devicefarm get-upload --arn arn:MyDefaultTestSpecARN
   ```

   Respons berisi URL yang telah ditetapkan sebelumnya tempat Anda dapat mengunduh spesifikasi pengujian default.
4. Contoh ini digunakan curl untuk mengunduh spesifikasi pengujian default dan menyimpannya sebagai`MyTestSpec.yml`:

   ```
   curl "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/ExampleURL" >
    MyTestSpec.yml
   ```
5. Anda dapat mengedit spesifikasi pengujian default untuk memenuhi persyaratan pengujian Anda, dan kemudian menggunakan spesifikasi pengujian yang dimodifikasi dalam uji coba di masa mendatang. Lewati langkah ini untuk menggunakan spesifikasi pengujian default apa adanya di lingkungan pengujian khusus.

6. Untuk membuat unggahan spesifikasi pengujian kustom Anda, jalankancreate-upload, tentukan nama spesifikasi pengujian Anda, jenis spesifikasi pengujian, dan ARN proyek.

   Contoh ini membuat unggahan untuk spesifikasi pengujian kustom Appium Java TestNG:

   ```
   aws devicefarm create-upload --name MyTestSpec.yml --type
    APPIUM_JAVA_TESTNG_TEST_SPEC --project-arn arn:MyProjectARN
   ```

   Tanggapan tersebut mencakup ARN unggahan spesifikasi pengujian dan URL yang telah ditentukan sebelumnya:

   ```
   {
       "upload": {
           "status": "INITIALIZED",
           "category": "PRIVATE",
           "name": "MyTestSpec.yml",
           "created": 1535751101.221,
           "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
   ExampleURL",
           "type": "APPIUM_JAVA_TESTNG_TEST_SPEC",
           "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
   c861-4c0a-b1d5-12345EXAMPLE"
       }
   }
   ```

7. Catat ARN untuk unggahan spesifikasi pengujian dan URL yang telah ditentukan sebelumnya.

8. Unggah file spesifikasi pengujian Anda menggunakan URL presigned Amazon S3. Contoh ini digunakan curl untuk mengunggah spesifikasi pengujian Appium JavaTest NG:

   ```
   curl -T MyTestSpec.yml "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
   ExampleURL"
   ```

9. Untuk memeriksa status unggahan spesifikasi pengujian Anda, jalankan get-upload dan tentukan ARN unggahan.

   ```
   aws devicefarm get-upload --arn arn:MyTestSpecUploadARN
   ```

   Tunggu hingga status dalam respons SUCCEEDED sebelum Anda menjadwalkan uji coba Anda.

   ```
   {
   ```

```
    "upload": {
        "status": "SUCCEEDED",
        "name": "MyTestSpec.yml",
        "created": 1535732625.964,
        "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL",
        "type": "APPIUM_JAVA_TESTNG_TEST_SPEC",
        "arn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE",
        "metadata": "{"valid": true}"
    }
}
```

Untuk memperbarui spesifikasi pengujian kustom Anda, jalankanupdate-upload, tentukan ARN unggahan untuk spesifikasi pengujian. Untuk informasi selengkapnya, lihat update-upload.

Langkah 6: Jadwalkan uji coba

Untuk menjadwalkan uji coba dengan AWS CLI, jalankanschedule-run, tentukan:

- Proyek ARN dari langkah 1.
- Perangkat mengumpulkan ARN dari langkah 2.
- Aplikasi mengunggah ARN dari langkah 3.
- Paket uji mengunggah ARN dari langkah 4.

Jika Anda menjalankan pengujian di lingkungan pengujian khusus, Anda juga memerlukan spesifikasi pengujian ARN dari langkah 5.

Untuk menjadwalkan lari di lingkungan pengujian standar

- Jalankanschedule-run, tentukan ARN proyek Anda, ARN kumpulan perangkat, ARN unggahan aplikasi, dan informasi paket uji.

  Contoh:

```
aws devicefarm schedule-run --project-arn arn:MyProjectARN --app-
arn arn:MyAppUploadARN --device-pool-arn arn:MyDevicePoolARN --name MyTestRun --
test type=APPIUM_JAVA_TESTNG,testPackageArn=arn:MyTestPackageARN
```

Respons berisi ARN run yang dapat Anda gunakan untuk memeriksa status uji coba Anda.


```
{
    "run": {
        "status": "SCHEDULING",
        "appUpload": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345appEXAMPLE",
        "name": "MyTestRun",
        "radios": {
            "gps": true,
            "wifi": true,
            "nfc": true,
            "bluetooth": true
        },
        "created": 1535756712.946,
        "totalJobs": 179,
        "completedJobs": 0,
        "platform": "ANDROID_APP",
        "result": "PENDING",
        "devicePoolArn": "arn:aws:devicefarm:us-
west-2:123456789101:devicepool:5e01a8c7-c861-4c0a-b1d5-12345devicepoolEXAMPLE",
        "jobTimeoutMinutes": 150,
        "billingMethod": "METERED",
        "type": "APPIUM_JAVA_TESTNG",
        "testSpecArn": "arn:aws:devicefarm:us-west-2:123456789101:upload:5e01a8c7-
c861-4c0a-b1d5-12345specEXAMPLE",
        "arn": "arn:aws:devicefarm:us-west-2:123456789101:run:5e01a8c7-c861-4c0a-
b1d5-12345runEXAMPLE",
        "counters": {
            "skipped": 0,
            "warned": 0,
            "failed": 0,
            "stopped": 0,
            "passed": 0,
            "errored": 0,
            "total": 0
        }
    }
}
```


Untuk informasi selengkapnya, lihat schedule-run.

Untuk menjadwalkan proses di lingkungan pengujian kustom

- Langkah-langkahnya hampir sama dengan langkah-langkahnya untuk lingkungan pengujian standar, dengan `testSpecArn` atribut tambahan dalam `--test` parameter.

  Contoh:

```
aws devicefarm schedule-run --project-arn arn:MyProjectARN --app-
arn arn:MyAppUploadARN --device-pool-arn arn:MyDevicePoolARN --name MyTestRun --
test
 testSpecArn=arn:MyTestSpecUploadARN,type=APPIUM_JAVA_TESTNG,testPackageArn=arn:MyTestPacka
```

Untuk memeriksa status uji coba Anda

- Gunakan get-run perintah dan tentukan run ARN:

```
aws devicefarm get-run --arn arn:aws:devicefarm:us-
west-2:111122223333:run:5e01a8c7-c861-4c0a-b1d5-12345runEXAMPLE
```

Untuk informasi selengkapnya, lihat get-run. Untuk informasi tentang menggunakan Device Farm dengan AWS CLI, lihatReferensi AWS CLI.

Buat uji coba (API)

Langkah-langkahnya sama dengan yang dijelaskan di AWS CLI bagian ini. Lihat Buat test run (AWS CLI).

Anda memerlukan informasi ini untuk memanggil `ScheduleRun`API:

- Sebuah proyek ARN. Lihat Buat proyek (API) dan `CreateProject`.
- Aplikasi mengunggah ARN. Lihat `CreateUpload`.
- Paket uji mengunggah ARN. Lihat `CreateUpload`.
- Perangkat kolam ARN. Lihat Buat kumpulan perangkat dan `CreateDevicePool`.

> **Note**
>
> Jika Anda menjalankan pengujian di lingkungan pengujian khusus, Anda juga memerlukan ARN unggah spesifikasi pengujian. Lihat informasi yang lebih lengkap di Langkah 5: (Opsional) Unggah spesifikasi pengujian khusus Anda dan `CreateUpload`.

Untuk informasi tentang menggunakan Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Langkah selanjutnya

Di konsol Device Farm, ikon jam

berubah menjadi ikon hasil seperti sukses

saat proses selesai. Laporan untuk proses muncul segera setelah pengujian selesai. Untuk informasi selengkapnya, lihat Laporan di AWS Device Farm.

Untuk menggunakan laporan, ikuti instruksi diBekerja dengan laporan pengujian di Device Farm.

Tetapkan batas waktu eksekusi untuk pengujian yang dijalankan di AWS Device Farm

Anda dapat menetapkan nilai berapa lama uji coba harus dijalankan sebelum menghentikan setiap perangkat menjalankan pengujian. Batas waktu eksekusi default adalah 150 menit per perangkat, tetapi Anda dapat menetapkan nilai serendah 5 menit. Anda dapat menggunakan konsol AWS Device Farm, AWS CLI, atau AWS Device Farm API untuk mengatur batas waktu eksekusi.

> **Important**
>
> Opsi batas waktu eksekusi harus diatur ke durasi maksimum untuk uji coba, bersama dengan beberapa buffer. Misalnya, jika pengujian Anda memakan waktu 20 menit per perangkat, Anda harus memilih batas waktu 30 menit per perangkat.

Jika eksekusi melebihi batas waktu Anda, eksekusi pada perangkat tersebut dihentikan secara paksa. Hasil sebagian tersedia, jika memungkinkan. Anda ditagih untuk eksekusi hingga saat itu,

jika Anda menggunakan opsi penagihan terukur. Untuk informasi selengkapnya tentang harga, lihat Harga Device Farm.

Anda mungkin ingin menggunakan fitur ini jika Anda tahu berapa lama waktu yang diperlukan untuk menjalankan uji coba di setiap perangkat. Saat menentukan batas waktu eksekusi untuk uji coba, Anda dapat menghindari situasi di mana uji coba macet karena alasan tertentu dan Anda ditagih untuk menit perangkat saat tidak ada pengujian yang dijalankan. Dengan kata lain, menggunakan fitur batas waktu eksekusi memungkinkan Anda menghentikan proses itu jika memakan waktu lebih lama dari yang diharapkan.

Anda dapat mengatur batas waktu eksekusi di dua tempat, di tingkat proyek dan tingkat uji coba.

Prasyarat

1. Selesaikan langkah-langkah dalam Menyiapkan.
2. Buat proyek di Device Farm. Ikuti instruksi diMembuat proyek di AWS Device Farm, dan kemudian kembali ke halaman ini.

Mengatur batas waktu eksekusi untuk sebuah proyek

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Jika Anda sudah memiliki proyek, pilih dari daftar. Jika tidak, pilih Proyek baru, masukkan nama untuk proyek Anda, lalu pilih Kirim.
4. Pilih Pengaturan proyek.
5. Pada tab Umum, untuk batas waktu Eksekusi, masukkan nilai atau gunakan bilah geser.
6. Pilih Simpan.

   Semua pengujian yang berjalan di proyek Anda sekarang menggunakan nilai batas waktu eksekusi yang Anda tentukan, kecuali jika Anda mengganti nilai batas waktu saat Anda menjadwalkan proses.

Mengatur batas waktu eksekusi untuk uji coba

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.

3. Jika Anda sudah memiliki proyek, pilih dari daftar. Jika tidak, pilih Proyek baru, masukkan nama untuk proyek Anda, lalu pilih Kirim.
4. Pilih Buat proses baru.
5. Ikuti langkah-langkah untuk memilih aplikasi, mengonfigurasi pengujian, memilih perangkat, dan menentukan status perangkat.
6. Pada Review dan mulai jalankan, untuk Setel batas waktu eksekusi, masukkan nilai atau gunakan bilah geser.
7. Pilih Konfirmasi dan mulai jalankan.

Simulasikan koneksi dan kondisi jaringan untuk AWS Device Farm berjalan

Anda dapat menggunakan pembentukan jaringan untuk mensimulasikan koneksi dan kondisi jaringan saat menguji aplikasi Android, iOS, FireOS, dan web Anda di Device Farm. Misalnya, Anda dapat menguji aplikasi dalam kondisi jaringan yang kurang sempurna.

Saat Anda membuat proses menggunakan pengaturan jaringan default, setiap perangkat memiliki koneksi Wi-Fi penuh tanpa hambatan dengan konektivitas internet. Saat Anda menggunakan pembentukan jaringan, Anda dapat mengubah koneksi Wi-Fi untuk menentukan profil jaringan seperti 3G atau Lossy WiFi yang mengontrol throughput, delay, jitter, dan loss untuk lalu lintas masuk dan keluar.

Topik

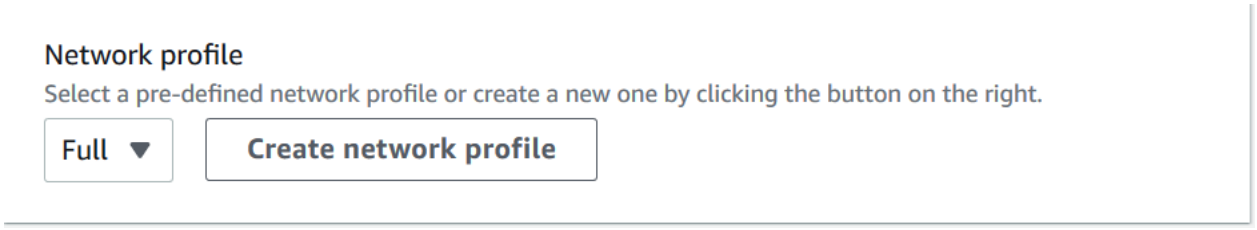
Siapkan pembentukan jaringan saat menjadwalkan uji coba

Saat Anda menjadwalkan proses, Anda dapat memilih dari salah satu profil Device Farm-curated, atau Anda dapat membuat dan mengelola profil Anda sendiri.

1. Dari proyek Device Farm apa pun, pilih Buat proses baru.

   Jika Anda belum memiliki proyek, lihatMembuat proyek di AWS Device Farm.

2. Pilih aplikasi Anda, lalu pilih Berikutnya.
3. Konfigurasikan pengujian Anda, lalu pilih Berikutnya.
4. Pilih perangkat Anda, lalu pilih Berikutnya.
5. Di bagian Pengaturan lokasi dan jaringan, pilih profil jaringan atau pilih Buat profil jaringan untuk membuat profil Anda sendiri.

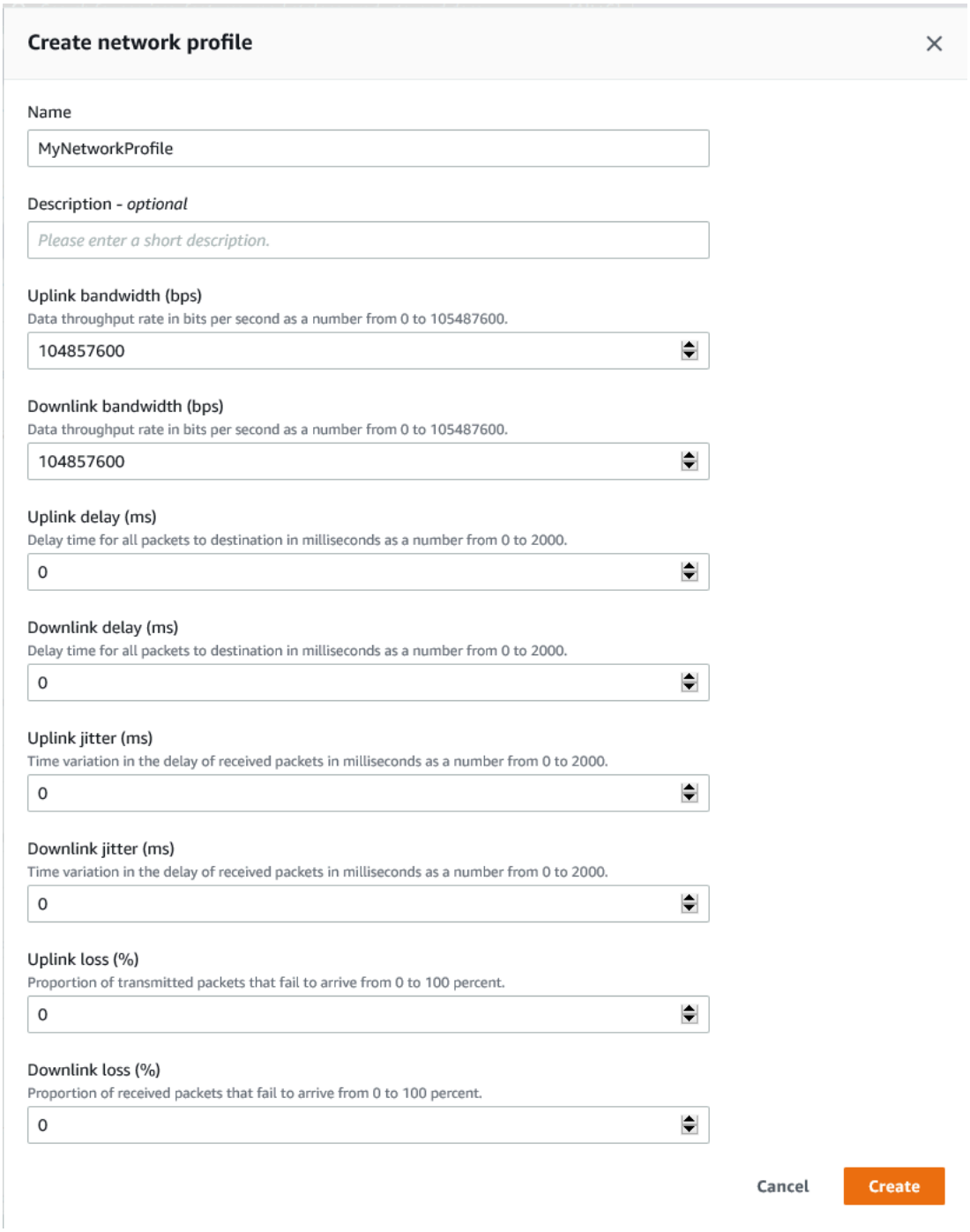


6. Pilih Berikutnya.
7. Tinjau dan mulai uji coba Anda.

Buat profil jaringan

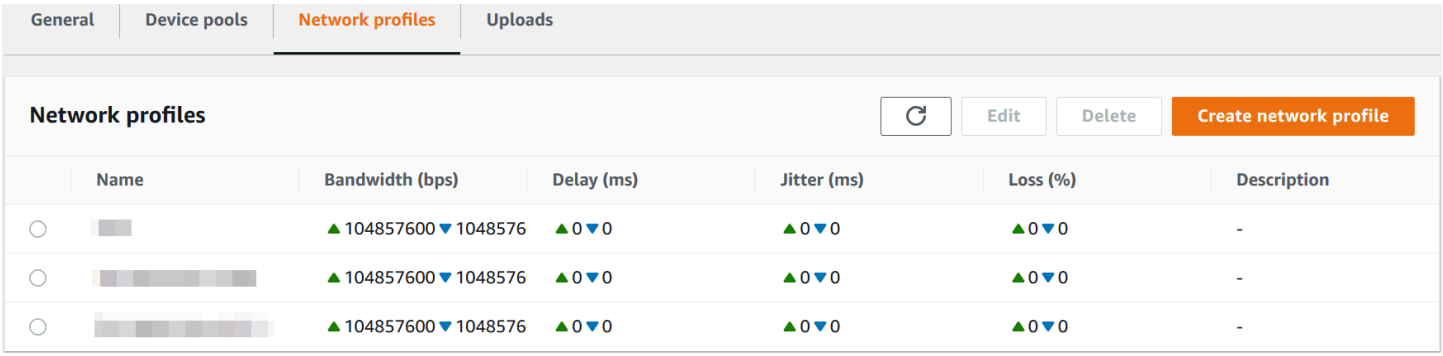
Saat Anda membuat uji coba, Anda dapat membuat profil jaringan.

1. Pilih Buat profil jaringan.



2. Masukkan nama dan pengaturan untuk profil jaringan Anda.
3. Pilih Buat.
4. Selesai membuat test run Anda dan mulai lari.

Setelah Anda membuat profil jaringan, Anda akan dapat melihat dan mengelolanya di halaman pengaturan Proyek.



Ubah kondisi jaringan selama pengujian

Anda dapat memanggil API dari host perangkat Anda menggunakan kerangka kerja seperti Appium untuk mensimulasikan kondisi jaringan dinamis seperti pengurangan bandwidth selama pengujian dijalankan. Untuk informasi lebih lanjut, lihat CreateNetworkProfile.

Hentikan proses di AWS Device Farm

Anda mungkin ingin berhenti berlari setelah Anda memulainya. Misalnya, jika Anda melihat masalah saat pengujian sedang berjalan, Anda mungkin ingin memulai ulang proses dengan skrip pengujian yang diperbarui.

Anda dapat menggunakan konsol Device Farm AWS CLI, atau API untuk menghentikan proses.

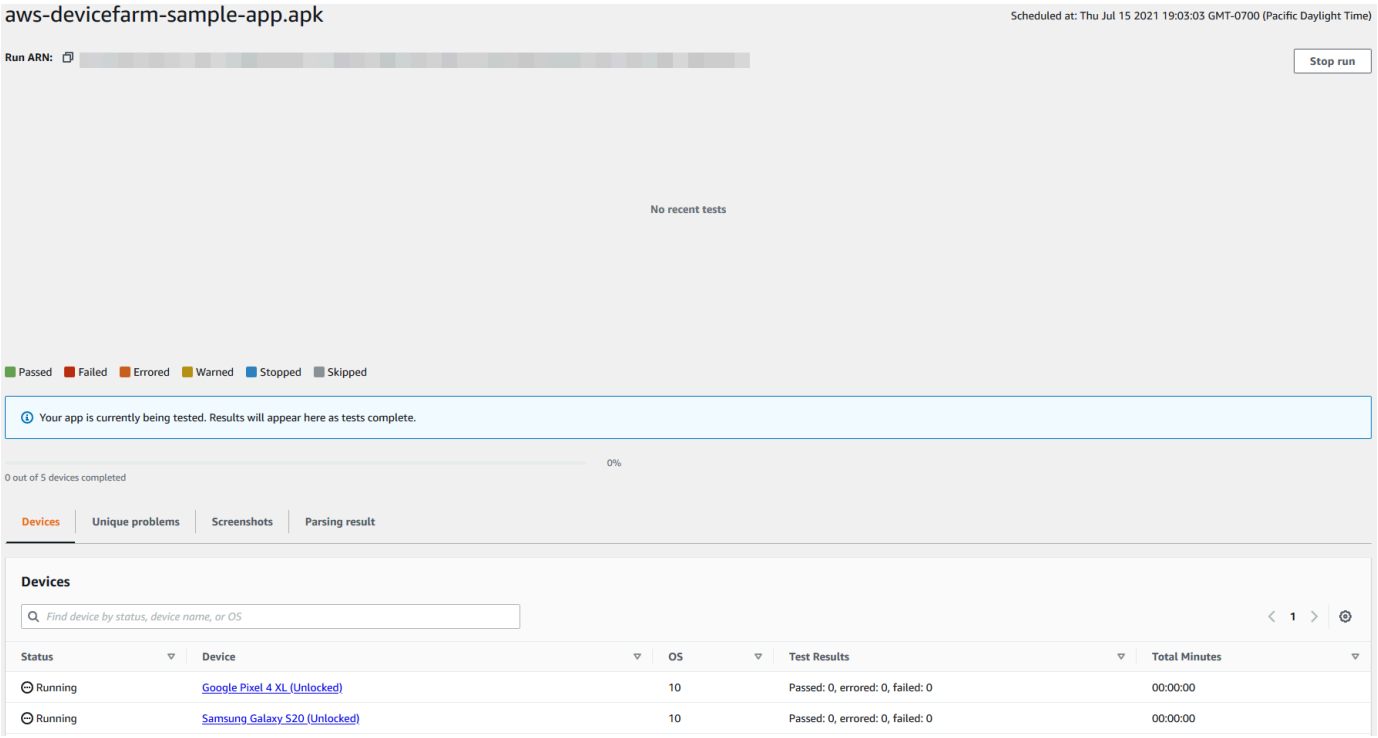
Topik

- Hentikan lari (konsol)
- Hentikan lari (AWS CLI)
- Hentikan lari (API)

Hentikan lari (konsol)

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Pilih proyek tempat Anda menjalankan uji coba aktif.
4. Pada halaman Pengujian otomatis, pilih uji coba.

   Ikon yang tertunda atau berjalan akan muncul di sebelah kiri nama perangkat.

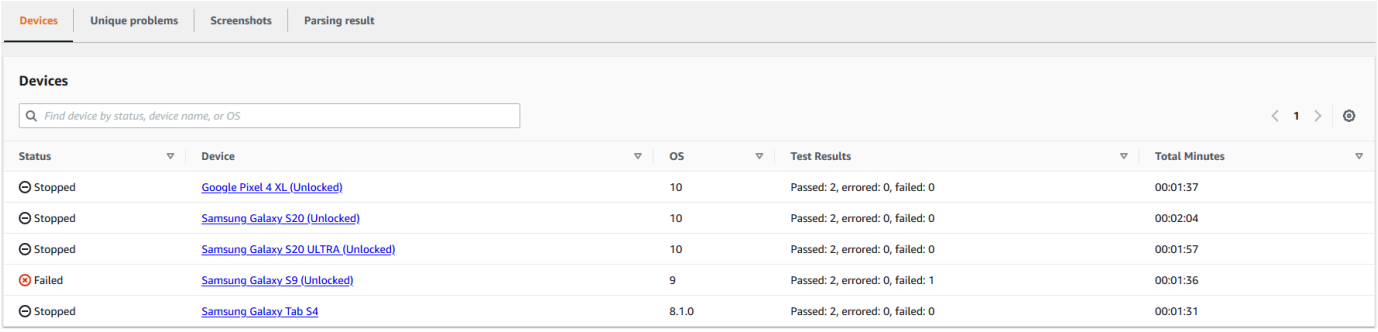


5. Pilih Stop run.

Setelah waktu yang singkat, ikon dengan lingkaran merah dengan minus di dalamnya muncul di sebelah nama perangkat. Ketika proses telah dihentikan, warna ikon berubah dari merah menjadi hitam.

> **Important**
>
> Jika pengujian telah dijalankan, Device Farm tidak dapat menghentikannya. Jika pengujian sedang berlangsung, Device Farm menghentikan pengujian. Total menit yang Anda akan ditagih muncul di bagian Perangkat. Selain itu, Anda juga akan ditagih untuk total menit yang dibutuhkan Device Farm untuk menjalankan setup suite dan teardown suite. Untuk informasi selengkapnya, lihat Harga Device Farm.

Gambar berikut menunjukkan contoh bagian Perangkat setelah uji coba berhasil dihentikan.



Hentikan lari (AWS CLI)

Anda dapat menjalankan perintah berikut untuk menghentikan uji coba yang ditentukan, di mana *mYarn* adalah Nama Sumber Daya Amazon (ARN) dari uji coba.

```
$ aws devicefarm stop-run --arn myARN
```

Anda akan melihat output yang serupa dengan yang berikut:

```
{
    "run": {
        "status": "STOPPING",
        "name": "Name of your run",
        "created": 1458329687.951,
        "totalJobs": 7,
        "completedJobs": 5,
        "deviceMinutes": {
            "unmetered": 0.0,
            "total": 0.0,
            "metered": 0.0
        },
        "platform": "ANDROID_APP",
        "result": "PENDING",
        "billingMethod": "METERED",
        "type": "BUILTIN_EXPLORER",
        "arn": "myARN",
        "counters": {
            "skipped": 0,
            "warned": 0,
            "failed": 0,
            "stopped": 0,
            "passed": 0,
```

```
            "errored": 0,
            "total": 0
        }
    }
}
```

Untuk mendapatkan ARN dari proses Anda, gunakan perintah. `list-runs` Output harus serupa dengan yang berikut ini:

```
{
    "runs": [
        {
            "status": "RUNNING",
            "name": "Name of your run",
            "created": 1458329687.951,
            "totalJobs": 7,
            "completedJobs": 5,
            "deviceMinutes": {
                "unmetered": 0.0,
                "total": 0.0,
                "metered": 0.0
            },
            "platform": "ANDROID_APP",
            "result": "PENDING",
            "billingMethod": "METERED",
            "type": "BUILTIN_EXPLORER",
            "arn": "Your ARN will be here",
            "counters": {
                "skipped": 0,
                "warned": 0,
                "failed": 0,
                "stopped": 0,
                "passed": 0,
                "errored": 0,
                "total": 0
            }
        }
    ]
}
```

Untuk informasi tentang menggunakan Device Farm dengan AWS CLI, lihatReferensi AWS CLI.

Hentikan lari (API)

- Panggil StopRunoperasi ke uji coba.

Untuk informasi tentang menggunakan Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Melihat daftar proses di AWS Device Farm

Anda dapat menggunakan konsol Device Farm AWS CLI, atau API untuk melihat daftar proses proyek.

Topik

- Lihat daftar proses (konsol)
- Lihat daftar run (AWS CLI)
- Melihat daftar run (API)

Lihat daftar proses (konsol)

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Dalam daftar proyek, pilih proyek yang sesuai dengan daftar yang ingin Anda lihat.

> **Tip**
>
> Anda dapat menggunakan bilah pencarian untuk memfilter daftar proyek berdasarkan nama.

Lihat daftar run (AWS CLI)

- Jalankan perintah list-runs.

  Untuk melihat informasi tentang satu proses, jalankan get-runperintah.

Untuk informasi tentang menggunakan Device Farm dengan AWS CLI, lihatReferensi AWS CLI.

Melihat daftar run (API)

- Panggil `ListRuns`API.

  Untuk melihat informasi tentang satu proses, panggil `GetRun`API.

Untuk informasi tentang Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Membuat kumpulan perangkat di AWS Device Farm

Anda dapat menggunakan konsol Device Farm AWS CLI, atau API untuk membuat kumpulan perangkat.

Topik

Prasyarat

- Buat run di konsol Device Farm. Ikuti petunjuk dalam Membuat uji coba di Device Farm. Saat Anda masuk ke halaman Pilih perangkat, lanjutkan dengan instruksi di bagian ini.

Buat kumpulan perangkat (konsol)

1. Pada halaman Pilih perangkat, pilih Buat kumpulan perangkat.
2. Untuk Nama, masukkan nama yang membuat kumpulan perangkat ini mudah diidentifikasi.
3. Untuk Deskripsi, masukkan deskripsi yang membuat kumpulan perangkat ini mudah diidentifikasi.
4. Jika Anda ingin menggunakan satu atau beberapa kriteria pemilihan untuk perangkat di kumpulan perangkat ini, lakukan hal berikut:

   a. Pilih Buat kumpulan perangkat dinamis.

   b. Pilih Tambahkan aturan.

c. Untuk Field (daftar drop-down pertama), pilih salah satu dari berikut ini:

   - Untuk menyertakan perangkat berdasarkan nama pabrikannya, pilih Produsen Perangkat.
   - Untuk menyertakan perangkat berdasarkan nilai tipenya, pilih Faktor Formulir.

d. Untuk Operator (daftar drop-down kedua), pilih EQUALS untuk menyertakan perangkat yang nilai Field sama dengan nilai Nilai.

e. Untuk Nilai (daftar drop-down ketiga), masukkan atau pilih nilai yang ingin Anda tentukan untuk nilai Bidang dan Operator. Jika Anda memilih Platform for Field, satu-satunya pilihan yang tersedia adalah ANDROID dan IOS. Demikian pula, jika Anda memilih Faktor Formulir untuk Bidang, satu-satunya pilihan yang tersedia adalah PHONE dan TABLET.

f. Untuk menambahkan aturan lain, pilih Tambahkan aturan.

g. Untuk menghapus aturan, pilih ikon X di sebelah aturan.

Setelah Anda membuat aturan pertama, dalam daftar perangkat, kotak di samping setiap perangkat yang cocok dengan aturan dipilih. Setelah Anda membuat atau mengubah aturan, dalam daftar perangkat, kotak di samping setiap perangkat yang cocok dengan aturan gabungan tersebut akan dipilih. Perangkat dengan kotak yang dipilih disertakan dalam kumpulan perangkat. Perangkat dengan kotak yang dibersihkan tidak termasuk.

5. Jika Anda ingin menyertakan atau mengecualikan perangkat individual secara manual, lakukan hal berikut:

   a. Pilih Buat kumpulan perangkat statis.

   b. Pilih atau kosongkan kotak di samping setiap perangkat. Anda dapat memilih atau menghapus kotak hanya jika Anda tidak memiliki aturan yang ditentukan.

6. Jika Anda ingin menyertakan atau mengecualikan semua perangkat yang ditampilkan, pilih atau kosongkan kotak di baris header kolom daftar.

> **Important**
>
> Meskipun Anda dapat menggunakan kotak di baris header kolom untuk mengubah daftar perangkat yang ditampilkan, ini tidak berarti bahwa perangkat yang ditampilkan yang tersisa adalah satu-satunya yang disertakan atau dikecualikan. Untuk mengonfirmasi perangkat mana yang disertakan atau dikecualikan, pastikan untuk menghapus konten semua kotak di baris header kolom, lalu telusuri kotaknya.

7. Pilih Buat.

Buat kumpulan perangkat (AWS CLI)

- Jalankan perintah create-device-pool.

Untuk informasi tentang menggunakan Device Farm dengan AWS CLI, lihatReferensi AWS CLI.

Membuat kumpulan perangkat (API)

- Panggil `CreateDevicePool`API.

Untuk informasi tentang menggunakan Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Menganalisis hasil di AWS Device Farm

Di lingkungan pengujian standar, Anda dapat menggunakan konsol Device Farm untuk melihat laporan untuk setiap pengujian dalam proses pengujian.

Device Farm juga mengumpulkan artefak lain seperti file, log, dan gambar yang dapat Anda unduh saat uji coba selesai.

Topik

- Bekerja dengan laporan pengujian di Device Farm
- Bekerja dengan artefak di Device Farm

Bekerja dengan laporan pengujian di Device Farm

Gunakan konsol Device Farm untuk melihat laporan pengujian Anda. Untuk informasi selengkapnya, lihat Laporan di AWS Device Farm.

Topik

- Prasyarat
- Memahami hasil tes
- Melihat laporan

Prasyarat

Siapkan uji coba dan verifikasi bahwa itu sudah selesai.

1. Untuk membuat run, lihatMembuat uji coba di Device Farm, dan kemudian kembali ke halaman ini.
2. Verifikasi bahwa proses selesai. Selama uji coba, konsol Device Farm menampilkan ikon tertunda

   untuk proses yang sedang berlangsung. Setiap perangkat yang sedang dijalankan juga akan dimulai dengan ikon yang tertunda, lalu beralih ke

   ikon yang sedang berjalan saat pengujian dimulai. Saat setiap pengujian selesai, ikon hasil pengujian ditampilkan di sebelah nama perangkat. Ketika semua pengujian telah selesai, ikon yang tertunda di sebelah run berubah menjadi ikon hasil pengujian. Untuk informasi selengkapnya, lihat Memahami hasil tes.

Memahami hasil tes

Konsol Device Farm menampilkan ikon yang membantu Anda menilai status uji coba selesai dengan cepat.

Topik

- Melaporkan hasil untuk tes individu
- Melaporkan hasil untuk beberapa tes

Melaporkan hasil untuk tes individu

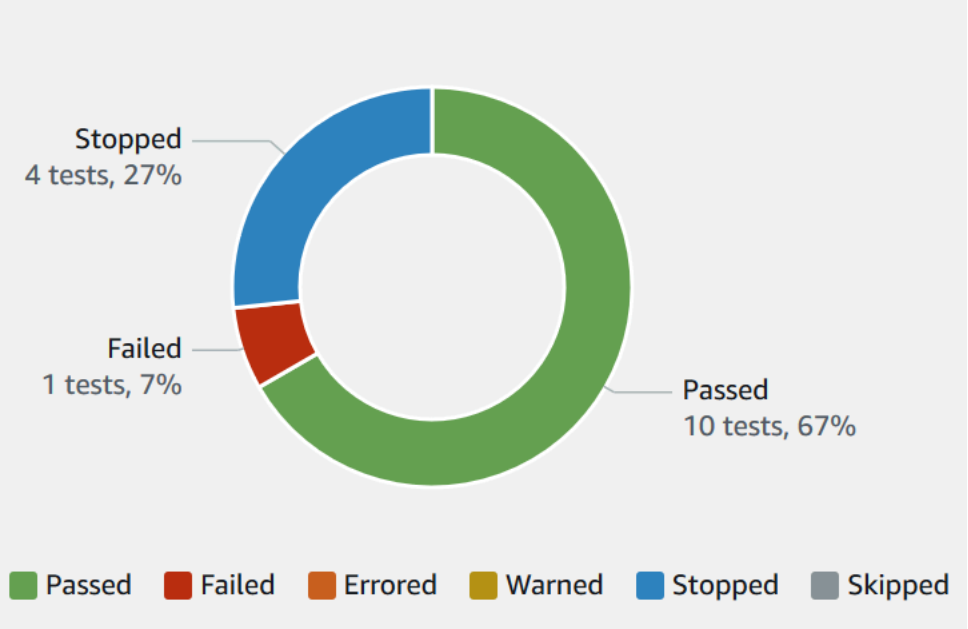
Untuk laporan yang menjelaskan pengujian individual, Device Farm menampilkan ikon:

Deskripsi	Ikon
Tes berhasil.	
Tes gagal.	

Deskripsi	Ikon
Device Farm melewatkan tes.	
Tes berhenti.	
Device Farm mengembalikan peringatan.	
Device Farm mengembalikan kesalahan.	

Melaporkan hasil untuk beberapa tes

Jika Anda memilih proses yang sudah selesai, Device Farm akan menampilkan grafik ringkasan hasil pengujian.



Misalnya, grafik hasil uji coba ini menunjukkan bahwa run memiliki 4 tes berhenti, 1 tes gagal, dan 10 tes yang berhasil.

Grafik selalu diberi kode warna dan diberi label.

Melihat laporan

Anda dapat melihat hasil pengujian di konsol Device Farm.

Topik

- Lihat halaman ringkasan uji coba
- Lihat laporan masalah unik
- Lihat laporan perangkat
- Lihat laporan rangkaian pengujian
- Lihat laporan pengujian
- Melihat data kinerja untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan
- Melihat informasi log untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan

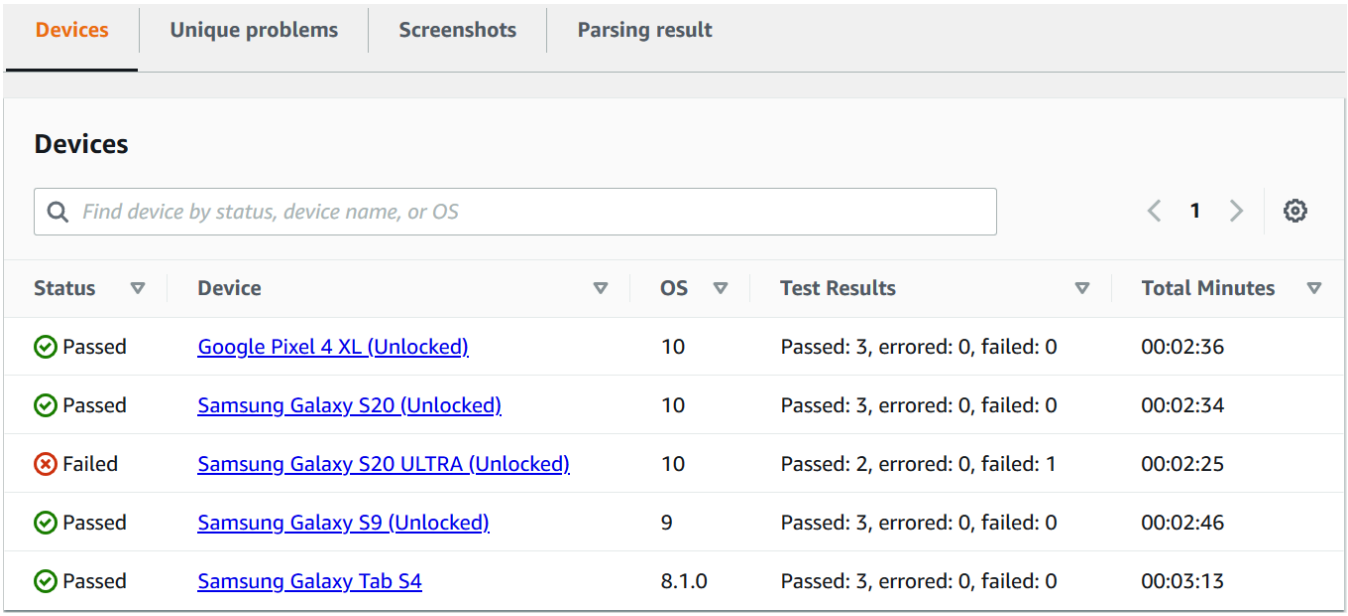
Lihat halaman ringkasan uji coba

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Di panel navigasi, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Dalam daftar proyek, pilih proyek untuk dijalankan.

   > **Tip**
   >
   > Untuk memfilter daftar proyek berdasarkan nama, gunakan bilah pencarian.

4. Pilih proses yang sudah selesai untuk melihat halaman laporan ringkasannya.
5. Halaman ringkasan uji coba menampilkan ikhtisar hasil pengujian Anda.

   - Bagian Masalah unik mencantumkan peringatan dan kegagalan unik. Untuk melihat masalah unik, ikuti instruksi diLihat laporan masalah unik.
   - Bagian Perangkat menampilkan jumlah total pengujian, berdasarkan hasil, untuk setiap perangkat.



Dalam contoh ini, ada beberapa perangkat. Pada entri tabel pertama, perangkat Google Pixel 4 XL yang menjalankan Android versi 10 melaporkan tiga pengujian yang berhasil yang membutuhkan waktu 02:36 menit untuk dijalankan.

Untuk melihat hasil berdasarkan perangkat, ikuti petunjuk diLihat laporan perangkat.

- Bagian Screenshots menampilkan daftar tangkapan layar apa pun yang ditangkap Device Farm selama proses dijalankan, dikelompokkan berdasarkan perangkat.
- Di bagian Hasil parsing, Anda dapat mengunduh hasil parsing.

Lihat laporan masalah unik

1. Dalam Masalah unik, pilih masalah yang ingin Anda lihat.
2. Pilih perangkat. Laporan menampilkan informasi tentang masalah tersebut.

   Bagian Video menampilkan rekaman video pengujian yang dapat diunduh.

   Bagian Hasil menampilkan hasil tes. Status direpresentasikan sebagai ikon hasil. Untuk informasi selengkapnya, lihat Melaporkan hasil untuk tes individu.

   Bagian Log menampilkan informasi apa pun yang dicatat oleh Device Farm selama pengujian. Untuk melihat informasi ini, ikuti instruksi diMelihat informasi log untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

Tab Performance menampilkan informasi tentang data performa apa pun yang dihasilkan Device Farm selama pengujian. Untuk melihat data kinerja ini, ikuti petunjuk diMelihat data kinerja untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

Tab File menampilkan daftar file terkait pengujian (seperti file log) yang dapat Anda unduh. Untuk mengunduh file, pilih tautan file dalam daftar.

Tab Screenshots menampilkan daftar tangkapan layar apa pun yang ditangkap Device Farm selama pengujian.

Lihat laporan perangkat

- Di bagian Perangkat, pilih perangkat.

  Bagian Video menampilkan rekaman video pengujian yang dapat diunduh.

  Bagian Suites menampilkan tabel yang berisi informasi tentang suite untuk perangkat.

  Dalam tabel ini, kolom Hasil pengujian merangkum jumlah pengujian berdasarkan hasil untuk setiap rangkaian pengujian yang telah berjalan di perangkat. Data ini juga memiliki komponen grafis. Untuk informasi selengkapnya, lihat Melaporkan hasil untuk beberapa tes.

  Untuk melihat hasil lengkap berdasarkan suite, ikuti petunjuk diLihat laporan rangkaian pengujian.

  Bagian Log menampilkan informasi apa pun yang dicatat oleh Device Farm untuk perangkat selama dijalankan. Untuk melihat informasi ini, ikuti instruksi diMelihat informasi log untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

  Bagian Performance menampilkan informasi tentang data performa apa pun yang dihasilkan Device Farm untuk perangkat selama dijalankan. Untuk melihat data kinerja ini, ikuti petunjuk diMelihat data kinerja untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

  Bagian File menampilkan daftar suite untuk perangkat dan file terkait apa pun (seperti file log) yang dapat Anda unduh. Untuk mengunduh file, pilih tautan file dalam daftar.

  Bagian Screenshots menampilkan daftar tangkapan layar apa pun yang ditangkap Device Farm selama menjalankan perangkat, dikelompokkan berdasarkan suite.

Lihat laporan rangkaian pengujian

1. Di bagian Perangkat, pilih perangkat.
2. Di bagian Suites, pilih suite dari tabel.

   Bagian Video menampilkan rekaman video pengujian yang dapat diunduh.

   Bagian Tes menampilkan tabel yang berisi informasi tentang pengujian di suite.

   Dalam tabel, kolom Hasil tes menampilkan hasilnya. Data ini juga memiliki komponen grafis. Untuk informasi selengkapnya, lihat Melaporkan hasil untuk beberapa tes.

   Untuk melihat hasil lengkap dengan tes, ikuti instruksi diLihat laporan pengujian.

   Bagian Log menampilkan informasi apa pun yang dicatat oleh Device Farm selama menjalankan suite. Untuk melihat informasi ini, ikuti instruksi diMelihat informasi log untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

   Bagian Performance menampilkan informasi tentang data performa apa pun yang dihasilkan Device Farm selama menjalankan suite. Untuk melihat data kinerja ini, ikuti petunjuk diMelihat data kinerja untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

   Bagian File menampilkan daftar pengujian untuk suite dan file terkait apa pun (seperti file log) yang dapat Anda unduh. Untuk mengunduh file, pilih tautan file dalam daftar.

   Bagian Screenshots menampilkan daftar tangkapan layar apa pun yang ditangkap Device Farm selama menjalankan suite, dikelompokkan berdasarkan pengujian.

Lihat laporan pengujian

1. Di bagian Perangkat, pilih perangkat.
2. Di bagian Suites, pilih suite.
3. Di bagian Tes, pilih tes.
4. Bagian Video menampilkan rekaman video pengujian yang dapat diunduh.

   Bagian Hasil menampilkan hasil tes. Status direpresentasikan sebagai ikon hasil. Untuk informasi selengkapnya, lihat Melaporkan hasil untuk tes individu.

Bagian Log menampilkan informasi apa pun yang dicatat oleh Device Farm selama pengujian. Untuk melihat informasi ini, ikuti instruksi diMelihat informasi log untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

Tab Performance menampilkan informasi tentang data performa apa pun yang dihasilkan Device Farm selama pengujian. Untuk melihat data kinerja ini, ikuti petunjuk diMelihat data kinerja untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan.

Tab File menampilkan daftar file terkait pengujian (seperti file log) yang dapat Anda unduh. Untuk mengunduh file, pilih tautan file dalam daftar.

Tab Screenshots menampilkan daftar tangkapan layar apa pun yang ditangkap Device Farm selama pengujian.

Melihat data kinerja untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan

> **Note**
>
> Device Farm mengumpulkan data performa perangkat hanya untuk perangkat Android saat ini.

Tab Performance menampilkan informasi berikut:

- Grafik CPU menampilkan persentase CPU yang digunakan aplikasi pada satu inti selama masalah, perangkat, rangkaian, atau pengujian yang dipilih (sepanjang sumbu vertikal) dari waktu ke waktu (sepanjang sumbu horizontal).

  Sumbu vertikal dinyatakan dalam persentase, dari 0% hingga persentase maksimum yang tercatat.

  Persentase ini mungkin melebihi 100% jika aplikasi menggunakan lebih dari satu inti. Misalnya, jika tiga core menggunakan 60%, persentase ini ditampilkan sebagai 180%.
- Grafik Memori menampilkan jumlah MB yang digunakan aplikasi selama masalah, perangkat, rangkaian, atau pengujian yang dipilih (sepanjang sumbu vertikal) dari waktu ke waktu (sepanjang sumbu horizontal).

  Sumbu vertikal dinyatakan dalam MB, dari 0 MB hingga jumlah maksimum MB yang direkam.

- Grafik Threads menampilkan jumlah utas yang digunakan selama masalah, perangkat, rangkaian, atau pengujian yang dipilih (sepanjang sumbu vertikal) dari waktu ke waktu (sepanjang sumbu horizontal).

  Sumbu vertikal dinyatakan dalam jumlah utas, dari nol utas hingga jumlah maksimum utas yang direkam.

Dalam semua kasus, sumbu horizontal diwakili, dalam hitungan detik, dari awal dan akhir proses untuk masalah, perangkat, suite, atau pengujian yang dipilih.

Untuk menampilkan informasi untuk titik data tertentu, jeda dalam grafik yang diinginkan pada detik yang diinginkan di sepanjang sumbu horizontal.

Melihat informasi log untuk masalah, perangkat, rangkaian, atau pengujian dalam laporan

Bagian Log menampilkan informasi berikut:

- Sumber mewakili sumber entri log. Nilai yang mungkin termasuk:
  - Harness mewakili entri log yang dibuat Device Farm. Entri log ini biasanya dibuat selama acara start dan stop.
  - Perangkat mewakili entri log yang dibuat perangkat. Untuk Android, entri log ini kompatibel dengan logcat-. Untuk iOS, entri log ini kompatibel dengan syslog.
  - Tes merupakan entri log yang dibuat oleh pengujian atau kerangka pengujian.
- Waktu mewakili waktu yang telah berlalu antara entri log pertama dan entri log ini. *Waktu dinyatakan dalam format MM:SS.SSS, di mana M mewakili menit dan S mewakili detik.*
- PID merupakan pengidentifikasi proses (PID) yang menciptakan entri log. Semua entri log yang dibuat oleh aplikasi pada perangkat memiliki PID yang sama.
- Level mewakili tingkat logging untuk entri log. Misalnya, `Logger.debug("This is a message!")` mencatat Level`Debug`. Ini adalah nilai yang mungkin:
  - Waspada
  - Kritis
  - Debug
  - Darurat
  - Kesalahan
  - Errored

- Failed
- Info
- Internal
- Pemberitahuan
- Lulus
- Dilewati
- Stopped
- Bertele-tele
- Diperingatkan
- Peringatan

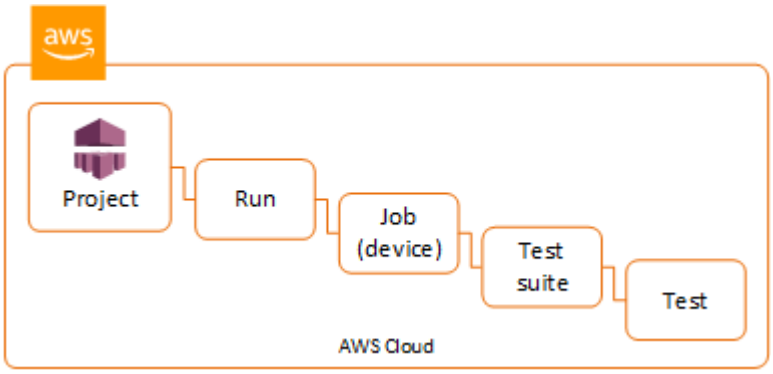
- Tag mewakili metadata arbitrer untuk entri log. Misalnya, Android logcat dapat menggunakan ini untuk menjelaskan bagian mana dari sistem yang membuat entri log (misalnya,`ActivityManager`).
- Pesan mewakili pesan atau data untuk entri log. Misalnya, `Logger.debug("Hello, World!")` mencatat Pesan dari`"Hello, World!"`.

Untuk menampilkan hanya sebagian dari informasi:

- Untuk menampilkan semua entri log yang cocok dengan nilai untuk kolom tertentu, masukkan nilai ke dalam bilah pencarian. Misalnya, untuk menampilkan semua entri log dengan nilai Sumber`Harness`, masukkan **`Harness`** di bilah pencarian.
- Untuk menghapus semua karakter dari kotak header kolom, pilih X di kotak header kolom itu. Menghapus semua karakter dari kotak header kolom sama dengan memasukkan ***** kotak header kolom itu.

Untuk mengunduh semua informasi log untuk perangkat, termasuk semua suite dan pengujian yang Anda jalankan, pilih Unduh log.

Bekerja dengan artefak di Device Farm



Device Farm mengumpulkan artefak seperti laporan, file log, dan gambar untuk setiap pengujian yang dijalankan.

Anda dapat mengunduh artefak yang dibuat selama uji coba:

Berkas

File yang dihasilkan selama uji coba termasuk laporan Device Farm. Untuk informasi selengkapnya, lihat Bekerja dengan laporan pengujian di Device Farm.

Log

Output dari setiap tes dalam uji coba.

Tangkapan layar

Gambar layar direkam untuk setiap pengujian dalam uji coba.

Menggunakan artefak (konsol)

1. Pada halaman laporan uji coba, dari Perangkat, pilih perangkat seluler.
2. Untuk mengunduh file, pilih salah satu dari File.
3. Untuk mengunduh log dari uji coba Anda, dari Log, pilih Unduh log.
4. Untuk mengunduh tangkapan layar, pilih tangkapan layar dari Screenshots.

Untuk informasi selengkapnya tentang mengunduh artefak di lingkungan pengujian khusus, lihatMenggunakan artefak di lingkungan pengujian khusus.

Menggunakan artefak ()AWS CLI

Anda dapat menggunakan daftar AWS CLI artefak uji coba Anda.

Topik

Langkah 1: Dapatkan Nama Sumber Daya Amazon Anda (ARN)

Anda dapat membuat daftar artefak Anda berdasarkan run, job, test suite, atau test. Anda membutuhkan ARN yang sesuai. Tabel ini menunjukkan input ARN untuk masing-masing perintah AWS CLI daftar:

AWS CLI Daftar Perintah	ARN yang dibutuhkan
list-projects	Perintah ini mengembalikan semua proyek dan tidak memerlukan ARN.
list-runs	`project`
list-jobs	`run`
list-suites	`job`
list-tests	`suite`

Misalnya, untuk menemukan ARN pengujian, jalankan list-tests menggunakan rangkaian pengujian ARN Anda sebagai parameter input.

Contoh:

```
aws devicefarm list-tests --arn arn:MyTestSuiteARN
```

Respons termasuk ARN tes untuk setiap tes dalam rangkaian pengujian.

```
{
```

```
    "tests": [
        {
            "status": "COMPLETED",
            "name": "Tests.FixturesTest.testExample",
            "created": 1537563725.116,
            "deviceMinutes": {
                "unmetered": 0.0,
                "total": 1.89,
                "metered": 1.89
            },
            "result": "PASSED",
            "message": "testExample passed",
            "arn": "arn:aws:devicefarm:us-west-2:123456789101:test:5e01a8c7-c861-4c0a-
b1d5-12345EXAMPLE",
            "counters": {
                "skipped": 0,
                "warned": 0,
                "failed": 0,
                "stopped": 0,
                "passed": 1,
                "errored": 0,
                "total": 1
            }
        }
    ]
}
```

Langkah 2: Daftar artefak Anda

Perintah AWS CLI daftar-artefak mengembalikan daftar artefak, seperti file, tangkapan layar, dan log. Setiap artefak memiliki URL sehingga Anda dapat mengunduh file.

- Panggilan list-artifacts yang menentukan ARN run, job, test suite, atau test. Tentukan jenis FILE, LOG, atau SCREENSHOT.

  Contoh ini mengembalikan URL unduhan untuk setiap artefak yang tersedia untuk pengujian individual:

  ```
  aws devicefarm list-artifacts --arn arn:MyTestARN --type "FILE"
  ```

  Respons berisi URL unduhan untuk setiap artefak.

```
{
    "artifacts": [
        {
            "url": "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/
ExampleURL",
            "extension": "txt",
            "type": "APPIUM_JAVA_OUTPUT",
            "name": "Appium Java Output",
            "arn": "arn:aws:devicefarm:us-west-2:123456789101:artifact:5e01a8c7-
c861-4c0a-b1d5-12345EXAMPLE",
        }
    ]
}
```

Langkah 3: Unduh artefak Anda

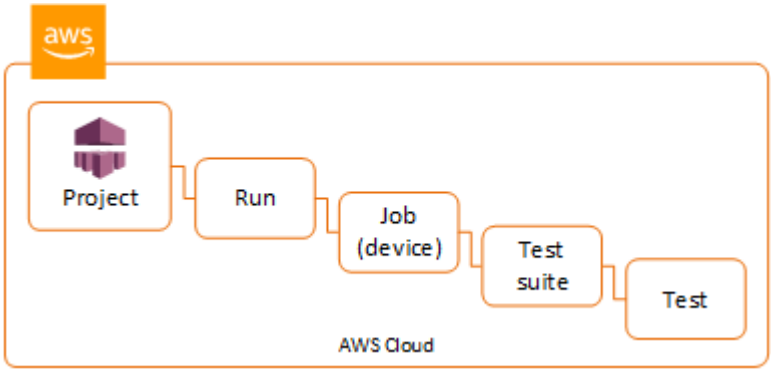
- Unduh artefak Anda menggunakan URL dari langkah sebelumnya. Contoh ini digunakan curl untuk mengunduh file keluaran Android Appium Java:

```
curl "https://prod-us-west-2-uploads.s3-us-west-2.amazonaws.com/ExampleURL"
 > MyArtifactName.txt
```

Menggunakan artefak (API)

ListArtifactsMetode Device Farm API menampilkan daftar artefak, seperti file, tangkapan layar, dan log. Setiap artefak memiliki URL sehingga Anda dapat mengunduh file.

Menggunakan artefak di lingkungan pengujian khusus



Dalam lingkungan pengujian khusus, Device Farm mengumpulkan artefak seperti laporan kustom, file log, dan gambar. Artefak ini tersedia untuk setiap perangkat dalam uji coba.

Anda dapat mengunduh artefak ini yang dibuat selama uji coba:

Uji keluaran spesifikasi

Output dari menjalankan perintah dalam file YAMAL spesifikasi pengujian.

Artefak pelanggan

File zip yang berisi artefak dari uji coba. Ini dikonfigurasi di bagian artefak: dari file YAMM spesifikasi pengujian Anda.

Uji skrip shell spesifikasi

File skrip shell perantara yang dibuat dari file YAMM Anda. Karena digunakan dalam uji coba, file skrip shell dapat digunakan untuk men-debug file YAMG.

Uji file spesifikasi

File YAMM yang digunakan dalam uji coba.

Untuk informasi selengkapnya, lihat Bekerja dengan artefak di Device Farm.

Menandai sumber daya AWS Device Farm

AWS Device Farm bekerja denganAWSAPI Penandaan Grup Sumber Daya. API ini memungkinkan Anda untuk mengelola sumber daya diAWSakun dengantag. Anda dapat menambahkan tag ke sumber daya, seperti proyek dan uji coba.

Anda dapat menggunakan tag untuk:

- Atur tagihan AWS Anda untuk mencerminkan struktur biaya Anda sendiri. Untuk melakukannya, daftar untuk mendapatkan tagihan akun AWS Anda dengan menyertakan nilai kunci tag. Lalu, untuk melihat biaya sumber daya gabungan, organisasikan informasi penagihan Anda sesuai dengan sumber daya Anda dengan nilai kunci tanda yang sama. Misalnya, Anda dapat menandai beberapa sumber daya dengan nama aplikasi, dan kemudian mengatur informasi penagihan Anda untuk melihat total biaya aplikasi tersebut di beberapa layanan. Untuk informasi selengkapnya, lihat Alokasi Biaya dan Pemberian Tanda di Tentang Manajemen Penagihan & Biaya AWS.
- Kontrol akses melalui kebijakan IAM. Untuk melakukannya, buat kebijakan yang memungkinkan akses ke sumber daya atau kumpulan sumber daya menggunakan kondisi nilai tag.
- Identifikasi dan kelola run yang memiliki properti tertentu sebagai tag, seperti cabang yang digunakan untuk pengujian.

Untuk informasi selengkapnya tentang penandaan sumber daya, lihatMenandai Praktik Terbaikwhitepaper.

Topik

- Penandaan sumber daya
- Mencari sumber daya berdasarkan tag
- Menghapus tag dari sumber daya

Penandaan sumber daya

AWS Resource Group Tagging API memungkinkan Anda menambahkan, menghapus, atau memodifikasi tag pada sumber daya. Untuk informasi lebih lanjut, lihatReferensi API Penandaan Grup Sumber Daya AWS.

Untuk menandai sumber daya, gunakan`TagResources`operasi dari`resourcegroupstaggingapi`titik akhir. Operasi ini mengambil daftar ARN dari layanan yang

didukung dan daftar pasangan nilai kunci. Nilai ini bersifat opsional. String kosong menunjukkan bahwa seharusnya tidak ada nilai untuk tag itu. Misalnya, contoh Python berikut menandai serangkaian ARN proyek dengan tag`build-config`dengan nilai`release`:

```
import boto3

client = boto3.client('resourcegroupstaggingapi')

client.tag_resources(ResourceARNList=["arn:aws:devicefarm:us-
west-2:111122223333:project:123e4567-e89b-12d3-a456-426655440000",
                                      "arn:aws:devicefarm:us-
west-2:111122223333:project:123e4567-e89b-12d3-a456-426655441111",
                                      "arn:aws:devicefarm:us-
west-2:111122223333:project:123e4567-e89b-12d3-a456-426655442222"]
                     Tags={"build-config":"release", "git-commit":"8fe28cb"})
```

Nilai tag tidak diperlukan. Untuk menyetel tag tanpa nilai, gunakan string kosong ("") saat menentukan nilai. Sebuah tag hanya dapat memiliki satu nilai. Nilai sebelumnya yang dimiliki tag untuk sumber daya akan ditimpa dengan nilai baru.

Mencari sumber daya berdasarkan tag

Untuk mencari sumber daya berdasarkan tag mereka, gunakan`GetResources`operasi dari`resourcegrouptaggingapi`titik akhir. Operasi ini mengambil serangkaian filter, tidak ada yang diperlukan, dan mengembalikan sumber daya yang sesuai dengan kriteria yang diberikan. Tanpa filter, semua sumber daya yang ditandai dikembalikan. The`GetResources`operasi memungkinkan Anda untuk memfilter sumber daya berdasarkan

- Nilai tanda
- Jenis sumber daya (misalnya,`devicefarm:run`)

Untuk informasi lebih lanjut, lihatReferensi API Penandaan Grup Sumber Daya AWS.

Contoh berikut mencari sesi pengujian browser desktop Device Farm (`devicefarm:testgrid-session`sumber daya) dengan tag`stack`yang memiliki nilai`production`:

```
import boto3
client = boto3.client('resourcegroupstaggingapi')
sessions = client.get_resources(ResourceTypeFilters=['devicefarm:testgrid-session'],
```

```
                                TagFilters=[
                                  {"Key":"stack","Values":["production"]}
                                ])
```

Menghapus tag dari sumber daya

Untuk menghapus tag, gunakan`UntagResources`operasi, menentukan daftar sumber daya dan tag untuk menghapus:

```
import boto3
client = boto3.client('resourcegroupstaggingapi')
client.UntagResources(ResourceARNList=["arn:aws:devicefarm:us-
west-2:111122223333:project:123e4567-e89b-12d3-a456-426655440000"], TagKeys=["RunCI"])
```

Bekerja dengan jenis pengujian di AWS Device Farm

Bagian ini menjelaskan dukungan Device Farm untuk kerangka kerja pengujian dan tipe pengujian bawaan.

Kerangka pengujian

Device Farm mendukung kerangka kerja pengujian otomatisasi seluler ini:

Kerangka kerja pengujian aplikasi Android

- Bekerja dengan Appium dan AWS Device Farm
- Bekerja dengan instrumentasi untuk Android dan AWS Device Farm

Kerangka kerja pengujian aplikasi iOS

- Bekerja dengan Appium dan AWS Device Farm
- Bekerja dengan XCTest untuk iOS dan AWS Device Farm
- XCTest UI

Kerangka kerja pengujian aplikasi web

Aplikasi web didukung menggunakan Appium. Untuk informasi lebih lanjut tentang membawa tes Anda ke Appium, lihat. Bekerja dengan Appium dan AWS Device Farm

Kerangka kerja di lingkungan pengujian khusus

Device Farm tidak menyediakan dukungan untuk menyesuaikan lingkungan pengujian untuk framework XCTest. Untuk informasi selengkapnya, lihat Bekerja dengan lingkungan pengujian khusus.

Dukungan versi Appium

Untuk pengujian yang berjalan di lingkungan khusus, Device Farm mendukung Appium versi 1. Untuk informasi selengkapnya, lihat Lingkungan uji.

Jenis pengujian bawaan

Dengan pengujian bawaan, Anda dapat menguji aplikasi di beberapa perangkat tanpa harus menulis dan memelihara skrip otomatisasi pengujian. Device Farm menawarkan satu jenis pengujian bawaan:

- Bawaan: fuzz (Android dan iOS)

Bekerja dengan Appium dan AWS Device Farm

Bagian ini menjelaskan cara mengonfigurasi, mengemas, dan mengunggah pengujian Appium Anda ke Device Farm. Appium adalah alat open source untuk mengotomatiskan aplikasi web asli dan seluler. Untuk informasi lebih lanjut, lihat Pengantar Appium di situs web Appium.

Untuk contoh aplikasi dan tautan ke pengujian yang berfungsi, lihat Aplikasi Sampel Device Farm untuk Android dan Aplikasi Sampel Device Farm untuk iOS aktif GitHub.

Dukungan versi

Support untuk berbagai framework dan bahasa pemrograman tergantung pada bahasa yang digunakan.

Device Farm mendukung semua versi server Appium 1.x dan 2.x. Untuk Android, Anda dapat memilih versi Appium utama dengan. `devicefarm-cli` Misalnya, untuk menggunakan server Appium versi 2, tambahkan perintah ini ke file YAMM spesifikasi pengujian Anda:

```
phases:
  install:
    commands:
      # To install a newer version of Appium such as version 2:
      - export APPIUM_VERSION=2
      - devicefarm-cli use appium $APPIUM_VERSION
```

Untuk iOS, Anda dapat memilih versi Appium tertentu dengan perintah `avm` ataunpm. Misalnya, untuk menggunakan `avm` perintah untuk menyetel versi server Appium ke 2.1.2, tambahkan perintah ini ke file YAMM spesifikasi pengujian Anda:

```
phases:
  install:
    commands:
      # To install a newer version of Appium such as version 2.1.2:
```

```
      - export APPIUM_VERSION=2.1.2
      - avm $APPIUM_VERSION
```

Menggunakan npm perintah untuk menggunakan versi terbaru Appium 2, tambahkan perintah ini ke file YAMM spesifikasi pengujian Anda:

```
phases:
  install:
    commands:
      - export APPIUM_VERSION=2
      - npm install -g appium@$APPIUM_VERSION
```

Untuk informasi selengkapnya tentang `devicefarm-cli` atau perintah CLI lainnya, lihat referensi AWS CLI.

Untuk menggunakan semua fitur kerangka kerja, seperti anotasi, pilih lingkungan pengujian khusus, dan gunakan AWS CLI atau `Device Farm` konsol untuk mengunggah spesifikasi pengujian khusus.

Topik

- Konfigurasikan paket pengujian Appium Anda
- Buat file paket uji zip
- Unggah paket pengujian Anda ke Device Farm
- Ambil tangkapan layar dari tes Anda (Opsional)

Konfigurasikan paket pengujian Appium Anda

Gunakan petunjuk berikut untuk mengonfigurasi paket pengujian Anda.

Java (JUnit)

1. Ubah `pom.xml` untuk mengatur kemasan ke file JAR:

   ```
   <groupId>com.acme</groupId>
   <artifactId>acme-myApp-appium</artifactId>
   <version>1.0-SNAPSHOT</version>
   <packaging>jar</packaging>
   ```

2. Ubah `pom.xml` `maven-jar-plugin` untuk digunakan untuk membangun pengujian Anda menjadi file JAR.

Plugin berikut membangun kode sumber pengujian Anda (apa pun di `src/test` direktori) ke dalam file JAR:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>2.6</version>
  <executions>
    <execution>
      <goals>
        <goal>test-jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

3. Ubah `pom.xml` untuk digunakan `maven-dependency-plugin` untuk membangun dependensi sebagai file JAR.

   Plugin berikut menyalin dependensi Anda ke direktori: `dependency-jars`

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-dependency-plugin</artifactId>
  <version>2.10</version>
  <executions>
    <execution>
      <id>copy-dependencies</id>
      <phase>package</phase>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <outputDirectory>${project.build.directory}/dependency-jars/</
outputDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>
```

4. Simpan rakitan XHTML berikut ke`src/main/assembly/zip.xml`.

XHTML berikut adalah definisi perakitan yang, ketika dikonfigurasi, menginstruksikan Maven untuk membuat file.zip yang berisi segala sesuatu di root direktori keluaran build Anda dan direktori: `dependency-jars`

```
<assembly
    xmlns="http://maven.apache.org/plugins/maven-assembly-plugin/assembly/1.1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-plugin/
assembly/1.1.0 http://maven.apache.org/xsd/assembly-1.1.0.xsd">
  <id>zip</id>
  <formats>
    <format>zip</format>
  </formats>
  <includeBaseDirectory>false</includeBaseDirectory>
  <fileSets>
    <fileSet>
      <directory>${project.build.directory}</directory>
      <outputDirectory>./</outputDirectory>
      <includes>
        <include>*.jar</include>
      </includes>
    </fileSet>
    <fileSet>
      <directory>${project.build.directory}</directory>
      <outputDirectory>./</outputDirectory>
      <includes>
        <include>/dependency-jars/</include>
      </includes>
    </fileSet>
  </fileSets>
</assembly>
```

5. Ubah `pom.xml` untuk digunakan `maven-assembly-plugin` untuk mengemas tes dan semua dependensi menjadi satu file.zip.

   Plugin berikut menggunakan rakitan sebelumnya untuk membuat file.zip bernama `zip-with-dependencies` di direktori keluaran build setiap kali mvn package dijalankan:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>2.5.4</version>
  <executions>
```

```
    <execution>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
      <configuration>
        <finalName>zip-with-dependencies</finalName>
        <appendAssemblyId>false</appendAssemblyId>
        <descriptors>
          <descriptor>src/main/assembly/zip.xml</descriptor>
        </descriptors>
      </configuration>
    </execution>
  </executions>
</plugin>
```

> **Note**
>
> Jika Anda menerima kesalahan yang mengatakan anotasi tidak didukung di 1.3, tambahkan yang berikut ini kepom.xml:
>
> ```
> <plugin>
>   <artifactId>maven-compiler-plugin</artifactId>
>   <configuration>
>     <source>1.7</source>
>     <target>1.7</target>
>   </configuration>
> </plugin>
> ```

Java (TestNG)

1. Ubah pom.xml untuk mengatur kemasan ke file JAR:

```
<groupId>com.acme</groupId>
<artifactId>acme-myApp-appium</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
```

2. Ubah `pom.xml` `maven-jar-plugin` untuk digunakan untuk membangun pengujian Anda menjadi file JAR.

   Plugin berikut membangun kode sumber pengujian Anda (apa pun di `src/test` direktori) ke dalam file JAR:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-jar-plugin</artifactId>
  <version>2.6</version>
  <executions>
    <execution>
      <goals>
        <goal>test-jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

3. Ubah `pom.xml` untuk digunakan `maven-dependency-plugin` untuk membangun dependensi sebagai file JAR.

   Plugin berikut menyalin dependensi Anda ke direktori: `dependency-jars`

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-dependency-plugin</artifactId>
  <version>2.10</version>
  <executions>
    <execution>
      <id>copy-dependencies</id>
      <phase>package</phase>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <outputDirectory>${project.build.directory}/dependency-jars/</
outputDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>
```

4. Simpan rakitan XHTML berikut kesrc/main/assembly/zip.xml.

   XHTML berikut adalah definisi perakitan yang, ketika dikonfigurasi, menginstruksikan Maven untuk membuat file.zip yang berisi segala sesuatu di root direktori keluaran build Anda dan direktori: dependency-jars

```
<assembly
    xmlns="http://maven.apache.org/plugins/maven-assembly-plugin/assembly/1.1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/plugins/maven-assembly-plugin/
assembly/1.1.0 http://maven.apache.org/xsd/assembly-1.1.0.xsd">
  <id>zip</id>
  <formats>
    <format>zip</format>
  </formats>
  <includeBaseDirectory>false</includeBaseDirectory>
  <fileSets>
    <fileSet>
      <directory>${project.build.directory}</directory>
      <outputDirectory>./</outputDirectory>
      <includes>
        <include>*.jar</include>
      </includes>
    </fileSet>
    <fileSet>
      <directory>${project.build.directory}</directory>
      <outputDirectory>./</outputDirectory>
      <includes>
        <include>/dependency-jars/</include>
      </includes>
    </fileSet>
  </fileSets>
</assembly>
```

5. Ubah pom.xml untuk digunakan maven-assembly-plugin untuk mengemas tes dan semua dependensi menjadi satu file.zip.

   Plugin berikut menggunakan rakitan sebelumnya untuk membuat file.zip bernama zip-with-dependencies di direktori keluaran build setiap kali mvn package dijalankan:

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
```

```
  <version>2.5.4</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>single</goal>
      </goals>
      <configuration>
        <finalName>zip-with-dependencies</finalName>
        <appendAssemblyId>false</appendAssemblyId>
        <descriptors>
          <descriptor>src/main/assembly/zip.xml</descriptor>
        </descriptors>
      </configuration>
    </execution>
  </executions>
</plugin>
```

> **Note**
>
> Jika Anda menerima kesalahan yang mengatakan anotasi tidak didukung di 1.3, tambahkan yang berikut ini kepom.xml:
>
> ```
> <plugin>
>   <artifactId>maven-compiler-plugin</artifactId>
>   <configuration>
>     <source>1.7</source>
>     <target>1.7</target>
>   </configuration>
> </plugin>
> ```

Node.JS

Untuk mengemas pengujian Appium Node.js dan mengunggahnya ke Device Farm, Anda harus menginstal yang berikut ini di mesin lokal Anda:

- Manajer Versi Node (nvm)

Gunakan alat ini saat Anda mengembangkan dan mengemas pengujian Anda sehingga dependensi yang tidak perlu tidak disertakan dalam paket pengujian Anda.

- Node.Js
- npm-bundle (diinstal secara global)

1. Verifikasi bahwa nvm ada

   ```
   command -v nvm
   ```

   Anda harus melihat `nvm` sebagai output.

   Untuk informasi lebih lanjut, lihat nvm di. GitHub

2. Jalankan perintah ini untuk menginstal Node.js:

   ```
   nvm install node
   ```

   Anda dapat menentukan versi tertentu dari Node.js:

   ```
   nvm install 11.4.0
   ```

3. Verifikasi bahwa versi Node yang benar sedang digunakan:

   ```
   node -v
   ```

4. Instal npm-bundle secara global:

   ```
   npm install -g npm-bundle
   ```

Python

1. Kami sangat menyarankan Anda menyiapkan virtualenv Python untuk mengembangkan dan mengemas pengujian sehingga dependensi yang tidak perlu tidak disertakan dalam paket aplikasi Anda.

   ```
   $ virtualenv workspace
   $ cd workspace
   ```

```
$ source bin/activate
```

> **Tip**
>
> - Jangan membuat virtualenv Python dengan `--system-site-packages` opsi, karena itu mewarisi paket dari direktori paket situs global Anda. Ini dapat mengakibatkan termasuk dependensi di lingkungan virtual Anda yang tidak diperlukan oleh pengujian Anda.
> - Anda juga harus memverifikasi bahwa pengujian Anda tidak menggunakan dependensi yang bergantung pada pustaka asli, karena pustaka asli ini mungkin tidak ada pada instance tempat pengujian ini dijalankan.

2. Instal py.test di lingkungan virtual Anda.

```
$ pip install pytest
```

3. Instal klien Appium Python di lingkungan virtual Anda.

```
$ pip install Appium-Python-Client
```

4. Kecuali Anda menentukan jalur yang berbeda dalam mode kustom, Device Farm mengharapkan pengujian Anda disimpan`tests/`. Anda dapat menggunakan `find` untuk menampilkan semua file di dalam folder:

```
$ find tests/
```

Konfirmasikan bahwa file-file ini berisi rangkaian pengujian yang ingin Anda jalankan di Device Farm

```
tests/
tests/my-first-tests.py
tests/my-second-tests/py
```

5. Jalankan perintah ini dari folder ruang kerja lingkungan virtual Anda untuk menampilkan daftar pengujian Anda tanpa menjalankannya.

```
$ py.test --collect-only tests/
```

Konfirmasikan bahwa output menunjukkan pengujian yang ingin Anda jalankan di Device Farm.

6. Bersihkan semua file yang di-cache di bawah folder tests/Anda:

```
$ find . -name '__pycache__' -type d -exec rm -r {} +
$ find . -name '*.pyc' -exec rm -f {} +
$ find . -name '*.pyo' -exec rm -f {} +
$ find . -name '*~' -exec rm -f {} +
```

7. Jalankan perintah berikut di ruang kerja Anda untuk menghasilkan file requirements.txt:

```
$ pip freeze > requirements.txt
```

Ruby

Untuk mengemas tes Appium Ruby Anda dan mengunggahnya ke Device Farm, Anda harus menginstal yang berikut ini di mesin lokal Anda:

- Manajer Versi Ruby (RVM)

  Gunakan alat baris perintah ini saat Anda mengembangkan dan mengemas pengujian Anda sehingga dependensi yang tidak perlu tidak disertakan dalam paket pengujian Anda.
- Ruby
- Bundler (Permata ini biasanya dipasang dengan Ruby.)

1. Instal kunci yang diperlukan, RVM, dan Ruby. Untuk petunjuk, lihat Menginstal RVM di situs web RVM.

   Setelah instalasi selesai, muat ulang terminal Anda dengan keluar dan kemudian masuk lagi.

   > **Note**
   >
   > RVM dimuat sebagai fungsi untuk shell bash saja.

2. Verifikasi bahwa rvm sudah terpasang dengan benar

```
command -v rvm
```

Anda harus melihat `rvm` sebagai output.

3. Jika Anda ingin menginstal versi Ruby tertentu, seperti *2.5.3*, jalankan perintah berikut:

```
rvm install ruby 2.5.3 --autolibs=0
```

Verifikasi bahwa Anda menggunakan versi Ruby yang diminta:

```
ruby -v
```

4. Konfigurasikan bundler untuk mengkompilasi paket untuk platform pengujian yang Anda inginkan:

```
bundle config specific_platform true
```

5. Perbarui file.lock Anda untuk menambahkan platform yang diperlukan untuk menjalankan pengujian.

   - Jika Anda mengompilasi pengujian untuk dijalankan di perangkat Android, jalankan perintah ini untuk mengonfigurasi Gemfile agar menggunakan dependensi untuk host pengujian Android:

     ```
     bundle lock --add-platform x86_64-linux
     ```

   - Jika Anda mengompilasi pengujian untuk dijalankan di perangkat iOS, jalankan perintah ini untuk mengonfigurasi Gemfile agar menggunakan dependensi untuk host uji iOS:

     ```
     bundle lock --add-platform x86_64-darwin
     ```

6. bundlerPermata biasanya dipasang secara default. Jika tidak, instal:

```
gem install bundler -v 2.3.26
```

Buat file paket uji zip

> **Warning**
>
> Di Device Farm, struktur folder file dalam paket pengujian zip Anda penting, dan beberapa alat arsip akan mengubah struktur file ZIP Anda secara implisit. Kami menyarankan Anda mengikuti utilitas baris perintah yang ditentukan di bawah ini daripada menggunakan utilitas arsip yang dibangun ke dalam pengelola file desktop lokal Anda (seperti Finder atau Windows Explorer).

Sekarang, bundel pengujian Anda untuk Device Farm.

Java (JUnit)

Bangun dan kemas pengujian Anda:

```
$ mvn clean package -DskipTests=true
```

File `zip-with-dependencies.zip` akan dibuat sebagai hasilnya. Ini adalah paket tes Anda.

Java (TestNG)

Bangun dan kemas pengujian Anda:

```
$ mvn clean package -DskipTests=true
```

File `zip-with-dependencies.zip` akan dibuat sebagai hasilnya. Ini adalah paket tes Anda.

Node.JS

1. Periksa proyek Anda.

   Pastikan Anda berada di direktori root proyek Anda. Anda dapat `package.json` melihat di direktori root.

2. Jalankan perintah ini untuk menginstal dependensi lokal Anda.

   ```
   npm install
   ```

   Perintah ini juga membuat `node_modules` folder di dalam direktori Anda saat ini.

> **Note**
>
> Pada titik ini, Anda harus dapat menjalankan pengujian Anda secara lokal.

3. Jalankan perintah ini untuk mengemas file di folder Anda saat ini ke dalam file*.tgz. File diberi nama menggunakan `name` properti di `package.json` file Anda.

```
npm-bundle
```

File tarball (.tgz) ini berisi semua kode dan dependensi Anda.

4. Jalankan perintah ini untuk menggabungkan tarball (*.tgz file) yang dihasilkan pada langkah sebelumnya ke dalam satu arsip zip:

```
zip -r MyTests.zip *.tgz
```

Ini adalah `MyTests.zip` file yang Anda unggah ke Device Farm dalam prosedur berikut.

Python

Python 2

Buat arsip paket Python yang diperlukan (disebut “wheelhouse”) menggunakan pip:

```
$ pip wheel --wheel-dir wheelhouse -r requirements.txt
```

Package ruang kemudi, pengujian, dan persyaratan pip Anda ke dalam arsip zip untuk Device Farm:

```
$ zip -r test_bundle.zip tests/ wheelhouse/ requirements.txt
```

Python 3

Package tes dan persyaratan pip Anda ke dalam file zip:

```
$ zip -r test_bundle.zip tests/ requirements.txt
```

Ruby

1. Jalankan perintah ini untuk membuat lingkungan Ruby virtual:

```
# myGemset is the name of your virtual Ruby environment
rvm gemset create myGemset
```

2. Jalankan perintah ini untuk menggunakan lingkungan yang baru saja Anda buat:

```
rvm gemset use myGemset
```

3. Periksa kode sumber Anda.

   Pastikan Anda berada di direktori root proyek Anda. Anda dapat `Gemfile` melihat di direktori root.

4. Jalankan perintah ini untuk menginstal dependensi lokal Anda dan semua permata dari: `Gemfile`

```
bundle install
```

> **Note**
>
> Pada titik ini, Anda harus dapat menjalankan pengujian Anda secara lokal. Gunakan perintah ini untuk menjalankan pengujian secara lokal:
>
> ```
> bundle exec $test_command
> ```

5. Package permata Anda di `vendor/cache` folder.

```
# This will copy all the .gem files needed to run your tests into the vendor/
cache directory
bundle package --all-platforms
```

6. Jalankan perintah berikut untuk menggabungkan kode sumber Anda, bersama dengan semua dependensi Anda, ke dalam satu arsip zip:

```
zip -r MyTests.zip Gemfile vendor/ $(any other source code directory files)
```

   Ini adalah `MyTests.zip` file yang Anda unggah ke Device Farm dalam prosedur berikut.

Unggah paket pengujian Anda ke Device Farm

Anda dapat menggunakan konsol Device Farm untuk mengunggah pengujian.

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Jika Anda adalah pengguna baru, pilih Proyek baru, masukkan nama untuk proyek, lalu pilih Kirim.

   Jika Anda sudah memiliki proyek, Anda dapat memilihnya untuk mengunggah tes Anda ke sana.
4. Buka project Anda, lalu pilih Create a new run.
5. Untuk pengujian Android dan iOS asli

   Pada halaman Pilih aplikasi, pilih Aplikasi Seluler, lalu pilih Pilih File untuk mengunggah paket yang dapat didistribusikan aplikasi Anda.

   > **Note**
   >
   > File harus berupa Android `.apk` atau iOS `.ipa`. Aplikasi iOS harus dibangun untuk perangkat nyata, bukan Simulator.

   Untuk pengujian aplikasi Web Seluler

   Pada halaman Pilih aplikasi, pilih Aplikasi Web.
6. Berikan tes Anda nama yang sesuai. Ini mungkin berisi kombinasi spasi atau tanda baca.
7. Pilih Berikutnya.
8. Pada halaman Configure, di bagian Setup test framework, pilih ***bahasa*** Appium, lalu Pilih File.
9. Jelajahi dan pilih file.zip yang berisi pengujian Anda. File.zip harus mengikuti format yang dijelaskan dalamKonfigurasikan paket pengujian Appium Anda.
10. Pilih Jalankan pengujian Anda di lingkungan khusus. Lingkungan eksekusi ini memungkinkan kontrol penuh atas penyiapan pengujian, pembongkaran, dan pemanggilan, serta memilih versi runtime dan server Appium tertentu. Anda dapat mengonfigurasi lingkungan kustom Anda melalui file spesifikasi pengujian. Untuk informasi selengkapnya, lihat Bekerja dengan lingkungan pengujian khusus di AWS Device Farm.

11. Pilih Berikutnya, lalu ikuti petunjuk untuk memilih perangkat dan mulai menjalankan. Untuk informasi selengkapnya, lihat Membuat uji coba di Device Farm.

> **Note**
>
> Device Farm tidak mengubah pengujian Appium.

Ambil tangkapan layar dari tes Anda (Opsional)

Anda dapat mengambil tangkapan layar sebagai bagian dari pengujian Anda.

Device Farm menyetel `DEVICEFARM_SCREENSHOT_PATH` properti ke jalur yang sepenuhnya memenuhi syarat pada sistem file lokal tempat Device Farm mengharapkan tangkapan layar Appium disimpan. Direktori khusus uji tempat tangkapan layar disimpan ditentukan saat runtime. Tangkapan layar ditarik ke laporan Device Farm Anda secara otomatis. Untuk melihat tangkapan layar, di konsol Device Farm, pilih bagian Screenshots.

Untuk informasi selengkapnya tentang pengambilan tangkapan layar dalam pengujian Appium, lihat Mengambil Screenshot di dokumentasi Appium API.

Bekerja dengan pengujian Android di AWS Device Farm

Device Farm menyediakan dukungan untuk beberapa jenis pengujian otomatisasi untuk perangkat Android, dan dua pengujian bawaan.

Kerangka kerja pengujian aplikasi Android

Tes berikut tersedia untuk perangkat Android.

- Bekerja dengan Appium dan AWS Device Farm
- Bekerja dengan instrumentasi untuk Android dan AWS Device Farm

Jenis pengujian bawaan untuk Android

Ada satu jenis pengujian bawaan yang tersedia untuk perangkat Android.

- Bawaan: fuzz (Android dan iOS)

Bekerja dengan instrumentasi untuk Android dan AWS Device Farm

Device Farm menyediakan dukungan untuk Instrumentasi (JUnit, Espresso, Robotium, atau pengujian berbasis Instrumentasi) untuk Android.

Device Farm juga menyediakan contoh aplikasi Android dan tautan ke pengujian yang berfungsi di tiga kerangka kerja otomatisasi Android, termasuk Instrumentation (Espresso). Aplikasi contoh Device Farm untuk Android tersedia untuk diunduh GitHub.

Topik

- Apa itu instrumentasi?
- Unggah pengujian instrumentasi Android Anda
- Mengambil tangkapan layar dalam pengujian instrumentasi Android
- Pertimbangan tambahan untuk pengujian instrumentasi Android
- Penguraian uji mode standar

Apa itu instrumentasi?

Instrumentasi Android memungkinkan Anda untuk memanggil metode callback dalam kode pengujian sehingga Anda dapat menjalankan siklus hidup komponen selangkah demi selangkah, seolah-olah Anda sedang men-debug komponen. Untuk informasi selengkapnya, lihat Pengujian instrumen di bagian Jenis dan lokasi pengujian pada dokumentasi Alat Developer Android.

Unggah pengujian instrumentasi Android Anda

Gunakan konsol Device Farm untuk mengunggah pengujian Anda.

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Dalam daftar proyek, pilih proyek yang ingin Anda unggah pengujian.

> **Tip**
>
> Anda dapat menggunakan bilah pencarian untuk memfilter daftar proyek berdasarkan nama.

Untuk membuat proyek, ikuti instruksi diMembuat proyek di AWS Device Farm.

4. Jika tombol Create a new run ditampilkan, pilih tombol tersebut.
5. Pada halaman Pilih aplikasi, pilih Pilih File.
6. Jelajahi dan pilih file aplikasi Android Anda. File harus berupa file.apk.
7. Pilih Berikutnya.
8. Pada halaman Configure, di bagian Setup test framework, pilih Instrumentation, lalu pilih Choose File.
9. Jelajahi dan pilih file.apk yang berisi pengujian Anda.
10. Pilih Berikutnya, lalu lengkapi instruksi yang tersisa untuk memilih perangkat dan mulai menjalankan.

Mengambil tangkapan layar dalam pengujian instrumentasi Android

Anda dapat mengambil tangkapan layar sebagai bagian dari pengujian Instrumentasi Android Anda.

Untuk mengambil tangkapan layar, hubungi salah satu metode berikut:

- Untuk Robotium, panggil `takeScreenShot` metode (misalnya,`solo.takeScreenShot();`).
- Untuk Spoon, panggil `screenshot` metode, misalnya:

```
Spoon.screenshot(activity, "initial_state");
/* Normal test code... */
Spoon.screenshot(activity, "after_login");
```

Selama uji coba, Device Farm mendapatkan tangkapan layar dari lokasi berikut di perangkat, jika ada, lalu menambahkannya ke laporan pengujian:

- `/sdcard/robotium-screenshots`
- `/sdcard/test-screenshots`
- `/sdcard/Download/spoon-screenshots/`*`test-class-name`*`/`*`test-method-name`*
- `/data/data/`*`application-package-name`*`/app_spoon-screenshots/`*`test-class-name`*`/`*`test-method-name`*

Pertimbangan tambahan untuk pengujian instrumentasi Android

Sistem Animasi

> Sesuai dokumentasi Android untuk pengujian Espresso, disarankan agar animasi sistem dimatikan saat menguji pada perangkat nyata. Device Farm secara otomatis menonaktifkan pengaturan Skala Animasi Jendela, Skala Animasi Transisi, dan Skala Durasi Animator saat dijalankan dengan runner pengujian instrumentasi android.support.test.runner.AndroidJUnitRunner.

Perekam Uji

> Device Farm mendukung framework, seperti Robotium, yang memiliki alat record-and-playback scripting.

Penguraian uji mode standar

Dalam mode standar run, Device Farm mem-parsing rangkaian pengujian Anda dan mengidentifikasi kelas pengujian unik dan metode yang akan dijalankan. Ini dilakukan melalui alat yang disebut Dex Test Parser.

Saat diberi file .apk instrumentasi Android sebagai input, parser mengembalikan nama metode pengujian yang memenuhi syarat sepenuhnya yang cocok dengan konvensi JUnit 3 dan JUnit 4.

Untuk menguji ini di lingkungan lokal:

1. Unduh `dex-test-parser`biner.
2. Jalankan perintah berikut untuk mendapatkan daftar metode pengujian yang akan berjalan di Device Farm:

```
java -jar parser.jar path/to/apk path/for/output
```

Bekerja dengan pengujian iOS di AWS Device Farm

Device Farm menyediakan dukungan untuk beberapa jenis pengujian otomatisasi untuk perangkat iOS, dan pengujian bawaan.

Kerangka kerja pengujian aplikasi iOS

Tes berikut tersedia untuk perangkat iOS.

- Bekerja dengan Appium dan AWS Device Farm
- Bekerja dengan XCTest untuk iOS dan AWS Device Farm
- XCTest UI

Jenis pengujian bawaan untuk iOS

Saat ini ada satu jenis pengujian bawaan yang tersedia untuk perangkat iOS.

- Bawaan: fuzz (Android dan iOS)

Bekerja dengan XCTest untuk iOS dan AWS Device Farm

Dengan Device Farm, Anda dapat menggunakan framework XCTest untuk menguji aplikasi di perangkat nyata. Untuk informasi selengkapnya tentang XCTest, lihat Dasar-dasar Pengujian dalam Pengujian dengan Xcode.

Untuk menjalankan pengujian, Anda membuat paket untuk uji coba, dan Anda mengunggah paket ini ke Device Farm.

Topik

- Membuat paket untuk menjalankan XCTest Anda
- Mengunggah paket untuk XCTest Anda dijalankan ke Device Farm

Membuat paket untuk menjalankan XCTest Anda

Untuk menguji aplikasi Anda dengan menggunakan framework XCTest, Device Farm memerlukan hal berikut:

- Paket aplikasi Anda sebagai `.ipa` file.
- Paket XCTest Anda sebagai file. `.zip`

Anda membuat paket-paket ini dengan menggunakan output build yang dihasilkan Xcode. Selesaikan langkah-langkah berikut untuk membuat paket sehingga Anda dapat mengunggahnya ke Device Farm.

Untuk menghasilkan output build untuk aplikasi Anda

1. Buka project aplikasi Anda di Xcode.
2. Di menu tarik-turun skema di toolbar Xcode, pilih Perangkat iOS Generik sebagai tujuan.
3. Di menu Produk, pilih Build For, lalu pilih Testing.

Untuk membuat paket aplikasi

1. Di navigator proyek di Xcode, di bawah Produk, buka menu kontekstual untuk file bernama. *app-project-name*`.app` Kemudian, pilih Tampilkan di Finder. Finder membuka folder bernama`Debug-iphoneos`, yang berisi output yang dihasilkan Xcode untuk build pengujian Anda. Folder ini termasuk `.app` file Anda.
2. Di Finder, buat folder baru, dan beri nama`Payload`.
3. Salin *app-project-name*`.app` file, dan tempel di `Payload` folder.
4. Buka menu kontekstual untuk `Payload` folder dan pilih Kompres “Payload”. Sebuah file bernama `Payload.zip` dibuat.
5. Ubah nama file dan ekstensi `Payload.zip` ke*app-project-name*`.ipa`.

   Pada langkah selanjutnya, Anda memberikan file ini ke Device Farm. Untuk membuat file lebih mudah ditemukan, Anda mungkin ingin memindahkannya ke lokasi lain, seperti desktop Anda.
6. Secara opsional, Anda dapat menghapus `Payload` folder dan `.app` file di dalamnya.

Untuk membuat paket XCTest

1. Di Finder, di `Debug-iphoneos` direktori, buka menu kontekstual untuk file tersebut. *app-project-name*`.app` Kemudian, pilih Show Package Contents.
2. Dalam isi paket, buka `Plugins` folder. Folder ini berisi file bernama*app-project-name*`.xctest`.
3. Buka menu kontekstual untuk file ini dan pilih Kompres "" ***app-project-name*`.xctest`**. Sebuah file bernama *app-project-name*`.xctest.zip` dibuat.

Pada langkah selanjutnya, Anda memberikan file ini ke Device Farm. Untuk membuat file lebih mudah ditemukan, Anda mungkin ingin memindahkannya ke lokasi lain, seperti desktop Anda.

Mengunggah paket untuk XCTest Anda dijalankan ke Device Farm

Gunakan konsol Device Farm untuk mengunggah paket untuk pengujian Anda.

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Jika Anda belum memiliki proyek, buat satu. Untuk langkah-langkah untuk membuat proyek, lihatMembuat proyek di AWS Device Farm.

   Jika tidak, pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Pilih proyek yang ingin Anda gunakan untuk menjalankan pengujian.
4. Pilih Buat proses baru.
5. Pada halaman Pilih aplikasi, pilih Aplikasi Seluler.
6. Pilih Pilih File.
7. Jelajahi `.ipa` file untuk aplikasi Anda dan unggah.

   > **Note**
   >
   > `.ipa`Paket Anda harus dibangun untuk pengujian.

8. Setelah unggahan selesai, pilih Berikutnya.
9. Pada halaman Configure, di bagian Setup test framework, pilih XCTest. Kemudian, pilih Pilih File.
10. Jelajahi `.zip` file yang berisi paket XCTest untuk aplikasi Anda dan unggah.
11. Setelah unggahan selesai, pilih Berikutnya.
12. Selesaikan langkah-langkah yang tersisa dalam proses pembuatan proyek. Anda akan memilih perangkat yang ingin Anda uji dan menentukan status perangkat.
13. Setelah Anda mengonfigurasi proses Anda, pada halaman Tinjau dan mulai jalankan, pilih Konfirmasi dan mulai jalankan.

    Device Farm menjalankan pengujian Anda dan menunjukkan hasilnya di konsol.

Bekerja dengan kerangka pengujian XCTest UI untuk iOS dan AWS Device Farm

Device Farm menyediakan dukungan untuk kerangka pengujian XCTest UI untuk iOS. Secara khusus, Device Farm mendukung pengujian UI XCTest yang ditulis dalam Objective-C dan Swift.

Topik

- Apa itu kerangka pengujian XCTest UI?
- Siapkan pengujian UI XCTest iOS Anda
- Unggah pengujian UI XCTest iOS Anda
- Mengambil tangkapan layar dalam tes iOS XCTest UI

Apa itu kerangka pengujian XCTest UI?

Kerangka kerja XCTest UI adalah kerangka pengujian baru yang diperkenalkan dengan Xcode 7. Kerangka kerja ini memperluas XCTest dengan kemampuan pengujian UI. Untuk informasi selengkapnya, lihat Pengujian Antarmuka Pengguna di Pustaka Pengembang iOS.

Siapkan pengujian UI XCTest iOS Anda

Bundel runner pengujian UI XCTest iOS Anda harus terkandung dalam file.ipa yang diformat dengan benar.

Untuk membuat file.ipa, letakkan bundel my-project-name UITest-Runner.app Anda di direktori Payload kosong. Selanjutnya, arsipkan direktori Payload ke dalam file.zip dan kemudian ubah ekstensi file ke.ipa. Bundel *UITest-runner.app diproduksi oleh Xcode saat Anda membangun proyek untuk pengujian. Ini dapat ditemukan di direktori Produk untuk proyek Anda.

Unggah pengujian UI XCTest iOS Anda

Gunakan konsol Device Farm untuk mengunggah pengujian Anda.

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Dalam daftar proyek, pilih proyek yang ingin Anda unggah pengujian.

> **Tip**
>
> Anda dapat menggunakan bilah pencarian untuk memfilter daftar proyek berdasarkan nama.
> Untuk membuat proyek, ikuti instruksi di Membuat proyek di AWS Device Farm

4. Jika tombol Create a new run ditampilkan, pilih tombol tersebut.
5. Pada halaman Pilih aplikasi, pilih Pilih File.
6. Jelajahi dan pilih file aplikasi iOS Anda. File harus berupa file.ipa.

> **Note**
>
> Pastikan file.ipa Anda dibuat untuk perangkat iOS dan bukan untuk simulator.

7. Pilih Berikutnya.
8. Pada halaman Configure, di bagian Setup test framework, pilih XCTest UI, lalu pilih Choose File.
9. Jelajahi dan pilih file.ipa yang berisi runner uji iOS XCTest UI Anda.
10. Pilih Berikutnya, lalu lengkapi instruksi yang tersisa untuk memilih perangkat yang akan menjalankan pengujian Anda dan mulai menjalankannya.

Mengambil tangkapan layar dalam tes iOS XCTest UI

Tes XCTest UI menangkap tangkapan layar secara otomatis untuk setiap langkah pengujian Anda. Tangkapan layar ini ditampilkan dalam laporan pengujian Device Farm Anda. Tidak diperlukan kode tambahan.

Bekerja dengan pengujian aplikasi web di AWS Device Farm

Device Farm menyediakan pengujian dengan Appium untuk aplikasi web. Untuk informasi selengkapnya tentang menyiapkan pengujian Appium di Device Farm, lihat. the section called “Appium”

Aturan untuk perangkat terukur dan tidak terukur

Pengukuran mengacu pada penagihan untuk perangkat. Secara default, perangkat Device Farm diukur dan Anda dikenakan biaya per menit setelah menit uji coba gratis habis. Anda juga dapat

memilih untuk membeli perangkat yang tidak diukur, yang memungkinkan pengujian tanpa batas dengan biaya bulanan tetap. Untuk informasi selengkapnya tentang harga, lihat Harga AWS Device Farm.

Jika Anda memilih untuk memulai proses dengan kumpulan perangkat yang berisi perangkat iOS dan Android, ada aturan untuk perangkat terukur dan tidak terukur. Misalnya, jika Anda memiliki lima perangkat Android yang tidak diukur dan lima perangkat iOS yang tidak diukur, pengujian web Anda berjalan menggunakan perangkat yang tidak diukur.

Berikut adalah contoh lain: Misalkan Anda memiliki lima perangkat Android yang tidak diukur dan 0 perangkat iOS yang tidak diukur. Jika Anda hanya memilih perangkat Android untuk menjalankan web, perangkat yang tidak diukur akan digunakan. Jika Anda memilih perangkat Android dan iOS untuk menjalankan web, metode penagihan diukur, dan perangkat yang tidak diukur tidak digunakan.

Bekerja dengan pengujian bawaan di AWS Device Farm

Device Farm menyediakan dukungan untuk jenis pengujian bawaan untuk perangkat Android dan iOS.

Jenis pengujian bawaan

Pengujian bawaan memungkinkan Anda untuk menguji aplikasi Anda tanpa menulis skrip.

- Bawaan: fuzz (Android dan iOS)

Bekerja dengan uji bulu halus bawaan untuk Device Farm

Device Farm menyediakan tipe uji fuzz bawaan.

Apa itu uji bulu halus bawaan?

Tes fuzz bawaan secara acak mengirimkan peristiwa antarmuka pengguna ke perangkat dan kemudian melaporkan hasilnya.

Gunakan tipe uji fuzz bawaan

Gunakan konsol Device Farm untuk menjalankan uji fuzz bawaan.

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.

2. Pada panel navigasi Device Farm, pilih Pengujian Perangkat Seluler, lalu pilih Proyek.
3. Dalam daftar proyek, pilih proyek tempat Anda ingin menjalankan uji fuzz bawaan.

> **Tip**
>
> Anda dapat menggunakan bilah pencarian untuk memfilter daftar proyek berdasarkan nama.
> Untuk membuat proyek, ikuti instruksi diMembuat proyek di AWS Device Farm.

4. Jika tombol Create a new run ditampilkan, pilih tombol tersebut.
5. Pada halaman Pilih aplikasi, pilih Pilih File.
6. Jelajahi dan pilih file aplikasi tempat Anda ingin menjalankan uji fuzz bawaan.
7. Pilih Berikutnya.
8. Pada halaman Konfigurasi, di bagian Setup Test Framework, pilih Built-in: Fuzz.
9. Jika salah satu pengaturan berikut muncul, Anda dapat menerima nilai default atau menentukan sendiri:

   - Jumlah peristiwa: Tentukan angka antara 1 dan 10.000, yang mewakili jumlah peristiwa antarmuka pengguna untuk pengujian fuzz yang akan dilakukan.
   - Event throttle: Tentukan angka antara 0 dan 1.000, mewakili jumlah milidetik untuk pengujian fuzz menunggu sebelum melakukan acara antarmuka pengguna berikutnya.
   - Biji pengacak: Tentukan nomor untuk uji fuzz yang akan digunakan untuk mengacak peristiwa antarmuka pengguna. Menentukan nomor yang sama untuk tes fuzz berikutnya memastikan urutan peristiwa yang identik.

10. Pilih Berikutnya, lalu lengkapi instruksi yang tersisa untuk memilih perangkat dan mulai menjalankan.

Bekerja dengan lingkungan pengujian khusus di AWS Device Farm

AWS Device Farm memungkinkan mengonfigurasi lingkungan khusus untuk pengujian otomatis (mode kustom), yang merupakan pendekatan yang disarankan untuk semua pengguna Device Farm. Untuk mempelajari lebih lanjut tentang lingkungan di Device Farm, lihat Lingkungan pengujian.

Manfaat Mode Kustom sebagai lawan dari Mode Standar meliputi:

- Eksekusi end-to-end pengujian yang lebih cepat: Paket pengujian tidak diuraikan untuk mendeteksi setiap pengujian di suite, menghindari overhead praprosesing/postprocessing.
- Log langsung dan streaming video: Log pengujian sisi klien dan video Anda disiarkan langsung saat menggunakan Mode Kustom. Fitur ini tidak tersedia dalam mode standar.
- Menangkap semua artefak: Pada host dan perangkat, Mode Kustom memungkinkan Anda untuk menangkap semua artefak pengujian. Ini mungkin tidak dimungkinkan dalam mode standar.
- Lingkungan lokal yang lebih konsisten dan dapat direplikasi: Ketika dalam Mode Standar, artefak akan disediakan untuk setiap pengujian individu secara terpisah, yang dapat bermanfaat dalam keadaan tertentu. Namun, lingkungan pengujian lokal Anda mungkin menyimpang dari konfigurasi asli karena Device Farm menangani setiap pengujian yang dijalankan secara berbeda.

  Sebaliknya, Mode Kustom memungkinkan Anda membuat lingkungan eksekusi pengujian Device Farm secara konsisten sesuai dengan lingkungan pengujian lokal Anda.

Lingkungan khusus dikonfigurasi menggunakan file spesifikasi pengujian (spesifikasi pengujian) yang diformat YAML. Device Farm menyediakan file spesifikasi pengujian default untuk setiap jenis pengujian yang didukung yang dapat digunakan sebagaimana adanya atau disesuaikan; kustomisasi seperti filter pengujian atau file konfigurasi dapat ditambahkan ke spesifikasi pengujian. Spesifikasi pengujian yang diedit dapat disimpan untuk uji coba di masa mendatang.

Untuk informasi selengkapnya, lihat Mengunggah Spesifikasi Uji Kustom Menggunakan dan. AWS CLI Membuat uji coba di Device Farm

Topik

- Sintaks spesifikasi uji
- Contoh spesifikasi uji
- Bekerja dengan lingkungan pengujian Amazon Linux 2 untuk pengujian Android

- Variabel-variabel lingkungan
- Memigrasi pengujian dari lingkungan pengujian standar ke lingkungan pengujian khusus
- Memperluas lingkungan pengujian khusus di Device Farm

Sintaks spesifikasi uji

Ini adalah struktur file spesifikasi pengujian YAMM:

```
version: 0.1

phases:
  install:
    commands:
      - command
      - command
  pre_test:
    commands:
      - command
      - command
  test:
    commands:
      - command
      - command
  post_test:
    commands:
      - command
      - command

artifacts:
  - location
  - location
```

Spesifikasi tes berisi yang berikut:

`version`

Mencerminkan versi spesifikasi pengujian yang didukung Device Farm. Nomor versi saat ini adalah 0.1.

`phases`

Bagian ini berisi kelompok perintah yang dijalankan selama uji coba.

Nama fase uji yang diizinkan adalah:

`install`

Opsional.

Dependensi default untuk kerangka kerja pengujian yang didukung oleh Device Farm sudah diinstal. Fase ini berisi perintah tambahan, jika ada, bahwa Device Farm berjalan selama instalasi.

`pre_test`

Opsional.

Perintah, jika ada, dijalankan sebelum pengujian otomatis Anda dijalankan.

`test`

Opsional.

Perintah dijalankan selama uji coba otomatis Anda dijalankan. Jika ada perintah dalam fase uji gagal, tes ditandai sebagai gagal.

`post_test`

Opsional.

Perintah, jika ada, dijalankan setelah pengujian otomatis Anda dijalankan.

`artifacts`

Opsional.

Device Farm mengumpulkan artefak seperti laporan kustom, file log, dan gambar dari lokasi yang ditentukan di sini. Karakter wildcard tidak didukung sebagai bagian dari lokasi artefak, jadi Anda harus menentukan jalur yang valid untuk setiap lokasi.

Artefak pengujian ini tersedia untuk setiap perangkat dalam uji coba Anda. Untuk informasi tentang mengambil artefak pengujian Anda, lihat. Menggunakan artefak di lingkungan pengujian khusus

> **Important**
>
> Spesifikasi pengujian harus diformat sebagai file YAMM yang valid. Jika indentasi atau spasi dalam spesifikasi pengujian Anda tidak valid, uji coba Anda bisa gagal. Tab tidak

diizinkan dalam file YAMM. Anda dapat menggunakan validator YAMM untuk menguji apakah spesifikasi pengujian Anda adalah file YAMM yang valid. Untuk informasi selengkapnya, lihat situs web YAMM.

Contoh spesifikasi uji

Ini adalah contoh spesifikasi pengujian YAMM Device Farm yang mengonfigurasi uji coba Appium Java TestNG:

```
version: 0.1

# This flag enables your test to run using Device Farm's Amazon Linux 2 test host when
 scheduled on
# Android devices. By default, iOS device tests will always run on Device Farm's macOS
 test hosts.
# For Android, you can explicitly select your test host to use our Amazon Linux 2
 infrastructure.
# For more information, please see:
# https://docs.aws.amazon.com/devicefarm/latest/developerguide/amazon-linux-2.html
android_test_host: amazon_linux_2

# Phases represent collections of commands that are executed during your test run on
 the test host.
phases:

  # The install phase contains commands for installing dependencies to run your tests.
  # For your convenience, certain dependencies are preinstalled on the test host.

  # For Android tests running on the Amazon Linux 2 test host, many software libraries
 are available
  # from the test host using the devicefarm-cli tool. To learn more, please see:
  # https://docs.aws.amazon.com/devicefarm/latest/developerguide/amazon-linux-2-
devicefarm-cli.html

  # For iOS tests, you can use the Node.JS tools nvm, npm, and avm to setup your
 environment. By
  # default, Node.js versions 16.20.2 and 14.19.3 are available on the test host.
  install:
    commands:
      # The Appium server is written using Node.js. In order to run your desired
 version of Appium,
```

```
    # you first need to set up a Node.js environment that is compatible with your
version of Appium.
    - |-
      if [ $DEVICEFARM_DEVICE_PLATFORM_NAME = "Android" ];
      then
        devicefarm-cli use node 16;
      else
        # For iOS, use "nvm use" to switch between the two preinstalled NodeJS
versions 14 and 16,
        # and use "nvm install" to download a new version of your choice.
        nvm use 16;
      fi;
    - node --version

    # Use the devicefarm-cli to select a preinstalled major version of Appium on
Android.
    # Use avm or npm to select Appium for iOS.
    - |-
      if [ $DEVICEFARM_DEVICE_PLATFORM_NAME = "Android" ];
      then
        # For Android, the Device Farm service automatically updates the preinstalled
Appium versions
        # over time to incorporate the latest minor and patch versions for each major
version. If you
        # wish to select a specific version of Appium, you can instead use NPM to
install it:
        # npm install -g appium@2.1.3;
        devicefarm-cli use appium 2;
      else
        # For iOS, Appium versions 1.22.2 and 2.2.1 are preinstalled and selectable
through avm.
        # For all other versions, please use npm to install them. For example:
        # npm install -g appium@2.1.3;
        # Note that, for iOS devices, Appium 2 is only supported on iOS version 14
and above using
        # NodeJS version 16 and above.
        avm 2.2.1;
      fi;
    - appium --version

    # For Appium version 2, for Android tests, Device Farm automatically updates the
preinstalled
    # UIAutomator2 driver over time to incorporate the latest minor and patch
versions for its major
```

```
      # version 2. If you want to install a specific version of the driver, you can use
the Appium
      # extension CLI to uninstall the existing UIAutomator2 driver and install your
desired version:
      # - |-
      #   if [ $DEVICEFARM_DEVICE_PLATFORM_NAME = "Android" ];
      #   then
      #     appium driver uninstall uiautomator2;
      #     appium driver install uiautomator2@2.34.0;
      #   fi;

      # For Appium version 2, for iOS tests, the XCUITest driver is preinstalled using
version 5.7.0
      # If you want to install a different version of the driver, you can use the
Appium extension CLI
      # to uninstall the existing XCUITest driver and install your desired version:
      # - |-
      #   if [ $DEVICEFARM_DEVICE_PLATFORM_NAME = "iOS" ];
      #   then
      #     appium driver uninstall xcuitest;
      #     appium driver install xcuitest@5.8.1;
      #   fi;

      # We recommend setting the Appium server's base path explicitly for accepting
commands.
      - export APPIUM_BASE_PATH=/wd/hub

      # Install the NodeJS dependencies.
      - cd $DEVICEFARM_TEST_PACKAGE_PATH
      # First, install dependencies which were packaged with the test package using
npm-bundle.
      - npm install *.tgz
      # Then, optionally, install any additional dependencies using npm install.
      # If you do run these commands, we strongly recommend that you include your
package-lock.json
      # file with your test package so that the dependencies installed on Device Farm
match
      # the dependencies you've installed locally.
      # - cd node_modules/*
      # - npm install

  # The pre-test phase contains commands for setting up your test environment.
  pre_test:
    commands:
```

```
      # Device farm provides different pre-built versions of WebDriverAgent, an
 essential Appium
      # dependency for iOS devices, and each version is suggested for different
 versions of Appium:
      # DEVICEFARM_WDA_DERIVED_DATA_PATH_V8: this version is suggested for Appium 2
      # DEVICEFARM_WDA_DERIVED_DATA_PATH_V7: this version is suggested for Appium 1
      # Additionally, for iOS versions 16 and below, the device unique identifier
 (UDID) needs
      # to be slightly modified for Appium tests.
      - |-
        if [ $DEVICEFARM_DEVICE_PLATFORM_NAME = "iOS" ];
        then
          if [ $(appium --version | cut -d "." -f1) -ge 2 ];
          then
            DEVICEFARM_WDA_DERIVED_DATA_PATH=$DEVICEFARM_WDA_DERIVED_DATA_PATH_V8;
          else
            DEVICEFARM_WDA_DERIVED_DATA_PATH=$DEVICEFARM_WDA_DERIVED_DATA_PATH_V7;
          fi;

          if [ $(echo $DEVICEFARM_DEVICE_OS_VERSION | cut -d "." -f 1) -le 16 ];
          then
            DEVICEFARM_DEVICE_UDID_FOR_APPIUM=$(echo $DEVICEFARM_DEVICE_UDID | tr -d
 "-");
          else
            DEVICEFARM_DEVICE_UDID_FOR_APPIUM=$DEVICEFARM_DEVICE_UDID;
          fi;
        fi;

      # Appium downloads Chromedriver using a feature that is considered insecure for
 multitenant
      # environments. This is not a problem for Device Farm because each test host is
 allocated
      # exclusively for one customer, then terminated entirely. For more information,
 please see
      # https://github.com/appium/appium/blob/master/packages/appium/docs/en/guides/
security.md

      # We recommend starting the Appium server process in the background using the
 command below.
      # The Appium server log will be written to the $DEVICEFARM_LOG_DIR directory.
      # The environment variables passed as capabilities to the server will be
 automatically assigned
      # during your test run based on your test's specific device.
```

```
      # For more information about which environment variables are set and how they're
 set, please see
      # https://docs.aws.amazon.com/devicefarm/latest/developerguide/custom-test-
environment-variables.html
      - |-
        if [ $DEVICEFARM_DEVICE_PLATFORM_NAME = "Android" ];
        then
          appium --base-path=$APPIUM_BASE_PATH --log-timestamp \
            --log-no-colors --relaxed-security --default-capabilities \
            "{\"appium:deviceName\": \"$DEVICEFARM_DEVICE_NAME\", \
            \"platformName\": \"$DEVICEFARM_DEVICE_PLATFORM_NAME\", \
            \"appium:app\": \"$DEVICEFARM_APP_PATH\", \
            \"appium:udid\":\"$DEVICEFARM_DEVICE_UDID\", \
            \"appium:platformVersion\": \"$DEVICEFARM_DEVICE_OS_VERSION\", \
            \"appium:chromedriverExecutableDir\":
 \"$DEVICEFARM_CHROMEDRIVER_EXECUTABLE_DIR\", \
            \"appium:automationName\": \"UiAutomator2\"}" \
            >> $DEVICEFARM_LOG_DIR/appium.log 2>&1 &
        else
          appium --base-path=$APPIUM_BASE_PATH --log-timestamp \
            --log-no-colors --relaxed-security --default-capabilities \
            "{\"appium:deviceName\": \"$DEVICEFARM_DEVICE_NAME\", \
            \"platformName\": \"$DEVICEFARM_DEVICE_PLATFORM_NAME\", \
            \"appium:app\": \"$DEVICEFARM_APP_PATH\", \
            \"appium:udid\":\"$DEVICEFARM_DEVICE_UDID_FOR_APPIUM\", \
            \"appium:platformVersion\": \"$DEVICEFARM_DEVICE_OS_VERSION\", \
            \"appium:derivedDataPath\": \"$DEVICEFARM_WDA_DERIVED_DATA_PATH\", \
            \"appium:usePrebuiltWDA\": true, \
            \"appium:automationName\": \"XCUITest\"}" \
            >> $DEVICEFARM_LOG_DIR/appium.log 2>&1 &
        fi;

      # This code will wait until the Appium server starts.
      - |-
        appium_initialization_time=0;
        until curl --silent --fail "http://0.0.0.0:4723${APPIUM_BASE_PATH}/status"; do
          if [[ $appium_initialization_time -gt 30 ]]; then
            echo "Appium did not start within 30 seconds. Exiting...";
            exit 1;
          fi;
          appium_initialization_time=$((appium_initialization_time + 1));
          echo "Waiting for Appium to start on port 4723...";
          sleep 1;
        done;
```

```
  # The test phase contains commands for running your tests.
  test:
    commands:
      # Your test package is downloaded and unpackaged into the
 $DEVICEFARM_TEST_PACKAGE_PATH directory.
      # When compiling with npm-bundle, the test folder can be found in the
 node_modules/*/ subdirectory.
      - cd $DEVICEFARM_TEST_PACKAGE_PATH/node_modules/*
      - echo "Starting the Appium NodeJS test"

      # Enter your command below to start the tests. The command should be the same
 command as the one
      # you use to run your tests locally from the command line. An example, "npm
 test", is given below:
      - npm test

  # The post-test phase contains commands that are run after your tests have completed.
  # If you need to run any commands to generating logs and reports on how your test
 performed,
  # we recommend adding them to this section.
  post_test:
    commands:

# Artifacts are a list of paths on the filesystem where you can store test output and
  reports.
# All files in these paths will be collected by Device Farm.
# These files will be available through the ListArtifacts API as your "Customer
  Artifacts".
artifacts:
  # By default, Device Farm will collect your artifacts from the $DEVICEFARM_LOG_DIR
 directory.
  - $DEVICEFARM_LOG_DIR
```

Bekerja dengan lingkungan pengujian Amazon Linux 2 untuk pengujian Android

AWS Device Farm menggunakan mesin host Amazon Elastic Compute Cloud (EC2) Amazon Elastic Compute Cloud (EC2) yang menjalankan Amazon Linux 2 untuk menjalankan pengujian Android. Saat Anda menjadwalkan uji coba, Device Farm mengalokasikan host khusus untuk setiap perangkat

untuk menjalankan pengujian secara independen. Mesin host berakhir setelah pengujian dijalankan bersama dengan artefak yang dihasilkan.

Host uji Amazon Linux 2 adalah lingkungan pengujian Android terbaru, menggantikan sistem berbasis Ubuntu sebelumnya. Dengan menggunakan file spesifikasi pengujian, Anda dapat memilih untuk menjalankan pengujian Android di lingkungan Amazon Linux 2.

Host Amazon Linux 2 memberikan beberapa keuntungan:

- Pengujian yang lebih cepat dan lebih andal: Dibandingkan dengan host lama, host uji baru secara signifikan meningkatkan kecepatan pengujian, terutama mengurangi waktu mulai pengujian. Host Amazon Linux 2 juga menunjukkan stabilitas dan keandalan yang lebih besar selama pengujian.
- Akses Jarak Jauh yang Ditingkatkan untuk pengujian manual: Peningkatan ke host pengujian terbaru dan peningkatan menghasilkan latensi yang lebih rendah dan kinerja video yang lebih baik untuk pengujian manual Android.
- Pemilihan versi perangkat lunak standar: Device Farm sekarang menstandarisasi dukungan bahasa pemrograman utama pada host uji serta versi kerangka Appium. Untuk bahasa yang didukung (saat ini Java, Python, Node.js, dan Ruby) dan Appium, host uji baru menyediakan rilis stabil jangka panjang segera setelah peluncuran. Manajemen versi terpusat melalui `devicefarm-cli` alat ini memungkinkan pengembangan file spesifikasi pengujian dengan pengalaman yang konsisten di seluruh kerangka kerja.

Topik

Perangkat lunak yang didukung

Host pengujian Amazon Linux 2 sudah diinstal sebelumnya dengan banyak pustaka perangkat lunak yang diperlukan untuk mendukung kerangka kerja pengujian Device Farm, menyediakan lingkungan pengujian siap saat diluncurkan. Untuk perangkat lunak lain yang diperlukan, Anda dapat memodifikasi file spesifikasi pengujian untuk diinstal dari paket pengujian Anda, mengunduh dari

internet, atau mengakses sumber pribadi dalam VPC Anda (lihat VPC ENI untuk informasi lebih lanjut). Untuk informasi selengkapnya, lihat contoh file spesifikasi Uji.

Versi perangkat lunak berikut saat ini tersedia di host:

Software Library	Software Version	Command to use in your test spec file
Python	3.8	`devicefarm-cli menggunakan python 3.8`
	3.9	`devicefarm-cli menggunakan python 3.9`
	3.10	`devicefarm-cli menggunakan python 3.10`
Java	8	`devicefarm-cli menggunakan java 8`
	11	`devicefarm-cli menggunakan java 11`
	17	`devicefarm-cli menggunakan java 17`
NodeJS	16	`devicefarm-cli menggunakan node 16`
	18	`devicefarm-cli menggunakan node 18`
Ruby	2.7	`devicefarm-cli gunakan ruby 2.7`
	3.2	`devicefarm-cli gunakan ruby 3.2`

Appium	1	`devicefarm-cli gunakan appium 1`
	2	`devicefarm-cli gunakan appium 2`

Host pengujian juga mencakup alat pendukung yang umum digunakan untuk setiap versi perangkat lunak, seperti manajer npm paket `pip` dan (disertakan dengan Python dan Node.js masing-masing) dan dependensi (seperti Driver Appium UIAutomator2) untuk alat seperti Appium. Ini memastikan Anda memiliki alat yang diperlukan untuk bekerja dengan kerangka kerja pengujian yang didukung.

`devicefarm-cli`Alat

Host uji Amazon Linux 2 menggunakan alat manajemen versi standar yang dipanggil `devicefarm-cli` untuk memilih versi perangkat lunak. Alat ini terpisah dari AWS CLI dan hanya tersedia di Device Farm Test Host. Dengan`devicefarm-cli`, Anda dapat beralih ke versi perangkat lunak yang sudah diinstal sebelumnya pada host uji. Ini memberikan cara mudah untuk memelihara file spesifikasi pengujian Device Farm Anda dari waktu ke waktu dan memberi Anda mekanisme yang dapat diprediksi untuk meningkatkan versi perangkat lunak di masa mendatang.

Cuplikan di bawah ini menunjukkan `help` halaman: `devicefarm-cli`

```
$ devicefarm-cli help
 Usage: devicefarm-cli COMMAND [ARGS]

     Commands:
         help                          Prints this usage message.
         list                          Lists all versions of software configurable
                                       via this CLI.
         use <software> <version>      Configures the software for usage within the
                                       current shell's environment.
```

Mari kita tinjau beberapa contoh menggunakan`devicefarm-cli`. Untuk menggunakan alat untuk mengubah versi Python dari *`3.10 ke 3.9`* dalam file spesifikasi pengujian Anda, jalankan perintah berikut:

```
$ python --version
Python 3.10.12
$ devicefarm-cli use python 3.9
```

```
$ python --version
Python 3.9.17
```

Untuk mengubah versi Appium dari 1 menjadi 2:

```
$ appium --version
1.22.3
$ devicefarm-cli use appium 2
$ appium --version
2.1.2
```

> **Tip**
>
> Perhatikan bahwa ketika Anda memilih versi perangkat lunak, `devicefarm-cli` juga beralih alat pendukung untuk bahasa-bahasa tersebut, seperti `pip` untuk Python dan `npm` NodeJS.

Pemilihan host uji Android

Untuk pengujian Android, Device Farm memerlukan bidang berikut dalam file spesifikasi pengujian Anda untuk memilih host pengujian Amazon Linux 2:

```
android_test_host: amazon_linux_2 | legacy
```

Gunakan `amazon_linux_2` untuk menjalankan pengujian Anda di host pengujian Amazon Linux 2:

```
android_test_host: amazon_linux_2
```

Pelajari lebih lanjut tentang manfaat Amazon Linux 2 di sini.

Device Farm merekomendasikan penggunaan host Amazon Linux 2 untuk pengujian Android alih-alih lingkungan host lama. Jika Anda lebih suka menggunakan lingkungan lama, gunakan `legacy` untuk menjalankan pengujian Anda di host pengujian lama:

```
android_test_host: legacy
```

Secara default, file spesifikasi pengujian tanpa pemilihan host pengujian akan berjalan di host uji lama.

Sintaks usang

Di bawah ini adalah sintaks usang untuk memilih Amazon Linux 2 di file spesifikasi pengujian Anda:

```
preview_features:
  android_amazon_linux_2_host: true
```

Jika Anda menggunakan flag ini, pengujian Anda akan terus berjalan di Amazon Linux 2. Namun, kami sangat menyarankan untuk menghapus bagian `preview_features` bendera dan menggantinya dengan `android_test_host` bidang baru untuk menghindari overhead pemeliharaan di masa mendatang.

> **Warning**
>
> Menggunakan kedua `android_amazon_linux_2_host` tanda `android_test_host` dan dalam file spesifikasi pengujian Anda akan mengembalikan kesalahan. Hanya satu yang harus digunakan; kami sarankan`android_test_host`.

Contoh file spesifikasi uji

Cuplikan berikut adalah contoh file spesifikasi pengujian Device Farm yang mengonfigurasi pengujian Appium NodeJS yang dijalankan menggunakan host pengujian Amazon Linux 2 untuk Android:

```
version: 0.1

# This flag enables your test to run using Device Farm's Amazon Linux 2 test host. For
 more information,
# please see https://docs.aws.amazon.com/devicefarm/latest/developerguide/amazon-
linux-2.html
android_test_host: amazon_linux_2

# Phases represent collections of commands that are executed during your test run on
 the test host.
phases:

  # The install phase contains commands for installing dependencies to run your tests.
  # For your convenience, certain dependencies are preinstalled on the test host. To
 lean about which
  # software is included with the host, and how to install additional software, please
 see:
```

```
  # https://docs.aws.amazon.com/devicefarm/latest/developerguide/amazon-linux-2-
supported-software.html

  # Many software libraries you may need are available from the test host using the
 devicefarm-cli tool.
  # To learn more about what software is available from it and how to use it, please
 see:
  # https://docs.aws.amazon.com/devicefarm/latest/developerguide/amazon-linux-2-
devicefarm-cli.html

  install:
    commands:
      # The Appium server is written using Node.js. In order to run your desired
 version of Appium,
      # you first need to set up a Node.js environment that is compatible with your
 version of Appium.
      - devicefarm-cli use node 18
      - node --version

      # Use the devicefarm-cli to select a preinstalled major version of Appium.
      - devicefarm-cli use appium 2
      - appium --version

      # The Device Farm service automatically updates the preinstalled Appium versions
 over time to
      # incorporate the latest minor and patch versions for each major version. If you
 wish to
      # select a specific version of Appium, you can use NPM to install it.
      # - npm install -g appium@2.1.3

      # For Appium version 2, Device Farm automatically updates the preinstalled
 UIAutomator2 driver
      # over time to incorporate the latest minor and patch versions for its major
 version 2. If you
      # want to install a specific version of the driver, you can use the Appium
 extension CLI to
      # uninstall the existing UIAutomator2 driver and install your desired version:
      # - appium driver uninstall uiautomator2
      # - appium driver install uiautomator2@2.34.0

      # We recommend setting the Appium server's base path explicitly for accepting
 commands.
      - export APPIUM_BASE_PATH=/wd/hub
```

```
      # Install the NodeJS dependencies.
      - cd $DEVICEFARM_TEST_PACKAGE_PATH
      # First, install dependencies which were packaged with the test package using
 npm-bundle.
      - npm install *.tgz
      # Then, optionally, install any additional dependencies using npm install.
      # If you do run these commands, we strongly recommend that you include your
 package-lock.json
      # file with your test package so that the dependencies installed on Device Farm
 match
      # the dependencies you've installed locally.
      # - cd node_modules/*
      # - npm install

  # The pre-test phase contains commands for setting up your test environment.
  pre_test:
    commands:

      # Appium downloads Chromedriver using a feature that is considered insecure for
 multitenant
      # environments. This is not a problem for Device Farm because each test host is
 allocated
      # exclusively for one customer, then terminated entirely. For more information,
 please see
      # https://github.com/appium/appium/blob/master/packages/appium/docs/en/guides/
security.md

      # We recommend starting the Appium server process in the background using the
 command below.
      # The Appium server log will be written to the $DEVICEFARM_LOG_DIR directory.
      # The environment variables passed as capabilities to the server will be
 automatically assigned
      # during your test run based on your test's specific device.
      # For more information about which environment variables are set and how they're
 set, please see
      # https://docs.aws.amazon.com/devicefarm/latest/developerguide/custom-test-
environment-variables.html
      - |-
        appium --base-path=$APPIUM_BASE_PATH --log-timestamp \
          --log-no-colors --relaxed-security --default-capabilities \
          "{\"appium:deviceName\": \"$DEVICEFARM_DEVICE_NAME\", \
          \"platformName\": \"$DEVICEFARM_DEVICE_PLATFORM_NAME\", \
          \"appium:app\": \"$DEVICEFARM_APP_PATH\", \
          \"appium:udid\":\"$DEVICEFARM_DEVICE_UDID\", \
```

```
          \"appium:platformVersion\": \"$DEVICEFARM_DEVICE_OS_VERSION\", \
          \"appium:chromedriverExecutableDir\":
 \"$DEVICEFARM_CHROMEDRIVER_EXECUTABLE_DIR\", \
          \"appium:automationName\": \"UiAutomator2\"}" \
          >> $DEVICEFARM_LOG_DIR/appium.log 2>&amp;1 &amp;

      # This code will wait until the Appium server starts.
      - |-
        appium_initialization_time=0;
        until curl --silent --fail "http://0.0.0.0:4723${APPIUM_BASE_PATH}/status"; do
          if [[ $appium_initialization_time -gt 30 ]]; then
            echo "Appium did not start within 30 seconds. Exiting...";
            exit 1;
          fi;
          appium_initialization_time=$((appium_initialization_time + 1));
          echo "Waiting for Appium to start on port 4723...";
          sleep 1;
        done;

  # The test phase contains commands for running your tests.
  test:
    commands:
      # Your test package is downloaded and unpackaged into the
 $DEVICEFARM_TEST_PACKAGE_PATH directory.
      # When compiling with npm-bundle, the test folder can be found in the
 node_modules/*/ subdirectory.
      - cd $DEVICEFARM_TEST_PACKAGE_PATH/node_modules/*
      - echo "Starting the Appium NodeJS test"

      # Enter your command below to start the tests. The command should be the same
 command as the one
      # you use to run your tests locally from the command line. An example, "npm
 test", is given below:
      - npm test

  # The post-test phase contains commands that are run after your tests have completed.
  # If you need to run any commands to generating logs and reports on how your test
 performed,
  # we recommend adding them to this section.
  post_test:
    commands:

# Artifacts are a list of paths on the filesystem where you can store test output and
 reports.
```

```
# All files in these paths will be collected by Device Farm.
# These files will be available through the ListArtifacts API as your "Customer
 Artifacts".
artifacts:
  # By default, Device Farm will collect your artifacts from the $DEVICEFARM_LOG_DIR
 directory.
  - $DEVICEFARM_LOG_DIR
```

Migrasi ke Host Uji Amazon Linux 2

Untuk memigrasikan pengujian yang ada dari host lama ke host Amazon Linux 2 yang baru, kembangkan file spesifikasi pengujian baru berdasarkan yang sudah ada sebelumnya. Pendekatan yang disarankan adalah memulai dengan file spesifikasi pengujian default baru untuk jenis pengujian Anda. Kemudian, migrasi perintah yang relevan dari file spesifikasi pengujian lama Anda ke yang baru, simpan file lama sebagai cadangan. Ini memungkinkan Anda memanfaatkan spesifikasi default yang dioptimalkan untuk host baru saat menggunakan kembali kode yang ada. Ini memastikan Anda mendapatkan manfaat penuh dari host baru yang dikonfigurasi secara optimal untuk pengujian Anda, sambil mempertahankan spesifikasi pengujian lama Anda untuk referensi saat Anda menyesuaikan perintah ke lingkungan baru.

Langkah-langkah berikut dapat digunakan untuk membuat file spesifikasi pengujian Amazon Linux 2 baru saat menggunakan kembali perintah dari file spesifikasi pengujian lama Anda:

1. Masuk ke konsol Device Farm di https://console.aws.amazon.com/devicefarm.
2. Arahkan ke proyek Device Farm yang berisi pengujian otomatisasi Anda.
3. Pilih Buat uji coba baru dalam proyek.
4. Pilih aplikasi dan paket pengujian yang sebelumnya digunakan untuk kerangka pengujian Anda.
5. Pilih Jalankan pengujian Anda di lingkungan khusus.
6. Pilih file spesifikasi pengujian yang saat ini Anda gunakan untuk pengujian pada host uji lama dari menu drop-down spesifikasi pengujian.
7. Salin isi file ini dan tempel secara lokal di editor teks untuk referensi nanti.
8. Di menu tarik-turun spesifikasi pengujian, ubah pilihan spesifikasi pengujian Anda ke file spesifikasi pengujian default terbaru.
9. Pilih Edit, dan Anda akan masuk ke antarmuka pengeditan spesifikasi pengujian. Anda akan melihat bahwa, di beberapa baris pertama file spesifikasi pengujian, file tersebut telah memilih host pengujian baru:

```
android_test_host: amazon_linux_2
```

10. Tinjau sintaks untuk memilih host uji di sini dan perbedaan utama antara host uji di sini.
11. Secara selektif menambahkan dan mengedit perintah dari file spesifikasi pengujian yang disimpan secara lokal dari langkah 6 ke file spesifikasi pengujian default yang baru. Kemudian, pilih Simpan sebagai untuk menyimpan file spesifikasi baru. Anda sekarang dapat menjadwalkan pengujian berjalan di host uji Amazon Linux 2.

Perbedaan antara host uji baru dan lama

Saat mengedit file spesifikasi pengujian untuk menggunakan host pengujian Amazon Linux 2 dan mentransisikan pengujian Anda dari host pengujian lama, perhatikan perbedaan lingkungan utama berikut:

- Memilih versi perangkat lunak: Dalam banyak kasus, versi perangkat lunak default telah berubah, jadi jika Anda tidak secara eksplisit memilih versi perangkat lunak Anda di host uji Legacy sebelumnya, Anda mungkin ingin menentukannya sekarang di host uji Amazon Linux 2 menggunakan. `devicefarm-cli` Dalam sebagian besar kasus penggunaan, kami menyarankan agar pelanggan secara eksplisit memilih versi perangkat lunak yang mereka gunakan. Dengan memilih versi perangkat lunak`devicefarm-cli`, Anda akan memiliki pengalaman yang dapat diprediksi dan konsisten dengannya dan menerima banyak peringatan jika Device Farm berencana untuk menghapus versi tersebut dari host pengujian.

  Selain itu, alat pemilihan perangkat lunak seperti `nvm` `pyenvavm`,,, dan `rvm` telah dihapus demi sistem pemilihan `devicefarm-cli` perangkat lunak baru.
- Versi perangkat lunak yang tersedia: Banyak versi perangkat lunak pra-instal sebelumnya telah dihapus, dan banyak versi baru telah ditambahkan. Jadi, pastikan bahwa ketika menggunakan `devicefarm-cli` untuk memilih versi perangkat lunak Anda, Anda memilih versi yang ada dalam daftar versi yang didukung.
- Setiap jalur file yang dikodekan keras dalam file spesifikasi pengujian host Legacy Anda sebagai jalur absolut kemungkinan besar tidak akan berfungsi seperti yang diharapkan di host pengujian Amazon Linux 2; mereka umumnya tidak direkomendasikan untuk penggunaan file spesifikasi pengujian. Kami menyarankan Anda menggunakan jalur relatif dan variabel lingkungan untuk semua kode file spesifikasi pengujian. Selain itu, perhatikan bahwa sebagian besar binari yang Anda butuhkan untuk pengujian dapat ditemukan di PATH host sehingga mereka segera dapat dijalankan dari file spesifikasi hanya dengan menggunakan namanya (seperti appium).

- Pengumpulan data kinerja tidak didukung pada host pengujian baru saat ini.
- Versi Sistem Operasi: Host uji warisan didasarkan pada sistem operasi Ubuntu, sedangkan yang baru didasarkan pada Amazon Linux 2. Akibatnya, pengguna mungkin melihat beberapa perbedaan dalam pustaka sistem yang tersedia dan versi pustaka sistem.
- Untuk pengguna Appium Java, host uji baru tidak berisi file JAR pra-instal di jalur kelasnya, sedangkan host sebelumnya berisi satu untuk kerangka TestNG (melalui variabel lingkungan). `$DEVICEFARM_TESTNG_JAR` Kami menyarankan agar pelanggan mengemas file JAR yang diperlukan untuk kerangka kerja pengujian mereka di dalam paket pengujian mereka dan menghapus instance `$DEVICEFARM_TESTNG_JAR` variabel dari file spesifikasi pengujian mereka. Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.
- Untuk pengguna Appium, variabel `$DEVICEFARM_CHROMEDRIVER_EXECUTABLE` lingkungan telah dihapus demi pendekatan baru untuk memungkinkan pelanggan mengakses Chromedriver untuk Android. Lihat file spesifikasi pengujian Default kami untuk contoh, yang menggunakan variabel `$DEVICEFARM_CHROMEDRIVER_EXECUTABLE_DIR` lingkungan baru.

> **Note**
>
> Kami sangat menyarankan untuk menjaga perintah server Appium yang ada dari file spesifikasi pengujian default apa adanya.

Kami merekomendasikan untuk menghubungi tim layanan melalui kasus dukungan jika Anda memiliki umpan balik atau pertanyaan tentang perbedaan antara host uji dari perspektif perangkat lunak.

Variabel-variabel lingkungan

Variabel lingkungan mewakili nilai yang digunakan oleh pengujian otomatis Anda. Anda dapat menggunakan variabel lingkungan ini dalam file YAMM dan kode pengujian Anda. Dalam lingkungan pengujian kustom, Device Farm secara dinamis mengisi variabel lingkungan saat runtime.

Topik

- Variabel lingkungan umum
- Variabel lingkungan Appium Java JUnit
- Variabel lingkungan Appium Java TestNG

- Variabel lingkungan XCUITest

Variabel lingkungan umum

Tes Android

Bagian ini menjelaskan variabel lingkungan khusus yang umum untuk pengujian platform Android yang didukung oleh Device Farm.

`$DEVICEFARM_DEVICE_NAME`

Nama perangkat tempat pengujian Anda dijalankan. Ini mewakili pengenal perangkat unik (UDID) perangkat.

`$DEVICEFARM_DEVICE_PLATFORM_NAME`

Nama platform perangkat. Ini adalah Android atau iOS.

`$DEVICEFARM_DEVICE_OS_VERSION`

Versi OS perangkat.

`$DEVICEFARM_APP_PATH`

Jalur ke aplikasi seluler di mesin host tempat pengujian dijalankan. Jalur aplikasi hanya tersedia untuk aplikasi seluler.

`$DEVICEFARM_DEVICE_UDID`

Pengidentifikasi unik perangkat seluler yang menjalankan pengujian otomatis.

`$DEVICEFARM_LOG_DIR`

Jalur ke file log yang dihasilkan selama uji coba. Secara default, semua file dalam direktori ini diarsipkan dalam file ZIP dan tersedia sebagai artefak setelah pengujian Anda dijalankan.

`$DEVICEFARM_SCREENSHOT_PATH`

Jalur ke tangkapan layar, jika ada, ditangkap selama uji coba.

`$DEVICEFARM_CHROMEDRIVER_EXECUTABLE_DIR`

Lokasi direktori yang berisi executable Chromedriver yang diperlukan untuk digunakan dalam tes web dan hybrid Appium.

$ANDROID_HOME

Jalur ke direktori instalasi Android SDK.

> **Note**
>
> Variabel `ANDROID_HOME` lingkungan hanya tersedia di host uji Amazon Linux 2 untuk Android.

Tes iOS

Bagian ini menjelaskan variabel lingkungan khusus yang umum untuk pengujian platform iOS yang didukung oleh Device Farm.

$DEVICEFARM_DEVICE_NAME

Nama perangkat tempat pengujian Anda dijalankan. Ini mewakili pengenal perangkat unik (UDID) perangkat.

$DEVICEFARM_DEVICE_PLATFORM_NAME

Nama platform perangkat. Ini adalah Android atau iOS.

$DEVICEFARM_APP_PATH

Jalur ke aplikasi seluler di mesin host tempat pengujian dijalankan. Jalur aplikasi hanya tersedia untuk aplikasi seluler.

$DEVICEFARM_DEVICE_UDID

Pengidentifikasi unik perangkat seluler yang menjalankan pengujian otomatis.

$DEVICEFARM_LOG_DIR

Jalur ke file log yang dihasilkan selama uji coba.

$DEVICEFARM_SCREENSHOT_PATH

Jalur ke tangkapan layar, jika ada, ditangkap selama uji coba.

Variabel lingkungan Appium Java JUnit

Bagian ini menjelaskan variabel lingkungan yang digunakan oleh tes JUnit Appium Java dalam lingkungan pengujian kustom.

$DEVICEFARM_TESTNG_JAR

Jalur ke file TestNg .jar.

$DEVICEFARM_TEST_PACKAGE_PATH

Jalur ke konten yang tidak di-zip dari file paket pengujian.

Variabel lingkungan Appium Java TestNG

Bagian ini menjelaskan variabel lingkungan yang digunakan oleh tes Appium Java TestNG di lingkungan pengujian kustom.

$DEVICEFARM_TESTNG_JAR

Jalur ke file TestNg .jar.

$DEVICEFARM_TEST_PACKAGE_PATH

Jalur ke konten yang tidak di-zip dari file paket pengujian.

Variabel lingkungan XCUITest

$DEVICEFARM_XCUITESTRUN_FILE

Jalur ke `.xctestun` file Device Farm. Ini dihasilkan dari aplikasi dan paket pengujian Anda.

$DEVICEFARM_DERIVED_DATA_PATH

Jalur yang diharapkan dari keluaran Device Farm xcodebuild.

Memigrasi pengujian dari lingkungan pengujian standar ke lingkungan pengujian khusus

Panduan berikut menjelaskan cara beralih dari mode eksekusi uji standar ke mode eksekusi kustom. Migrasi terutama melibatkan dua bentuk eksekusi yang berbeda:

1. Mode standar: Mode eksekusi pengujian ini terutama dibangun untuk menyediakan pelaporan terperinci dan lingkungan yang dikelola sepenuhnya kepada pelanggan.

2. Mode kustom: Mode eksekusi pengujian ini dibuat untuk berbagai kasus penggunaan yang memerlukan uji coba lebih cepat, kemampuan untuk mengangkat dan menggeser dan mencapai paritas dengan lingkungan lokal mereka, dan streaming video langsung.

Pertimbangan saat bermigrasi

Bagian ini mencantumkan beberapa kasus penggunaan yang menonjol untuk dipertimbangkan saat bermigrasi ke mode kustom:

1. Kecepatan: Dalam mode eksekusi standar, Device Farm mem-parsing metadata pengujian yang telah dikemas dan diunggah menggunakan instruksi pengemasan untuk kerangka kerja khusus Anda. Parsing mendeteksi jumlah tes dalam paket Anda. Setelah itu, Device Farm menjalankan setiap pengujian secara terpisah dan menyajikan log, video, dan artefak hasil lainnya secara individual untuk setiap pengujian. Namun, ini terus menambah total waktu eksekusi end-to-end pengujian karena ada pra dan pasca pemrosesan pengujian dan artefak hasil pada akhir layanan.

   Sebaliknya, mode eksekusi kustom tidak mengurai paket pengujian Anda; ini berarti tidak ada pra-pemrosesan dan pasca-pemrosesan minimal untuk pengujian atau artefak hasil. Ini menghasilkan total waktu end-to-end eksekusi dekat dengan pengaturan lokal Anda. Pengujian dijalankan dalam format yang sama seperti jika dijalankan pada mesin lokal Anda. Hasil tes sama dengan apa yang Anda dapatkan secara lokal dan tersedia untuk diunduh di akhir pelaksanaan pekerjaan.
2. Kustomisasi atau Fleksibilitas: Mode eksekusi standar mem-parsing paket pengujian Anda untuk mendeteksi jumlah pengujian dan kemudian menjalankan setiap pengujian secara terpisah. Perhatikan bahwa tidak ada jaminan bahwa tes akan berjalan sesuai urutan yang Anda tentukan. Akibatnya, tes yang membutuhkan urutan eksekusi tertentu mungkin tidak berfungsi seperti yang diharapkan. Selain itu, tidak ada cara untuk menyesuaikan lingkungan mesin host atau meneruskan file konfigurasi yang mungkin diperlukan untuk menjalankan pengujian Anda dengan cara tertentu.

   Sebaliknya, mode kustom memungkinkan Anda mengonfigurasi lingkungan mesin host termasuk kemampuan untuk menginstal perangkat lunak tambahan, meneruskan filter ke pengujian Anda, meneruskan file konfigurasi, dan mengontrol pengaturan eksekusi pengujian. Ini mencapai ini melalui file yaml (juga disebut file testspec) yang dapat Anda modifikasi dengan menambahkan perintah shell ke dalamnya. File yaml ini akan dikonversi ke skrip shell yang dieksekusi pada mesin host uji. Anda dapat menyimpan beberapa file yaml dan memilih satu secara dinamis sesuai kebutuhan Anda saat Anda menjadwalkan proses.

3. Video langsung dan logging: Mode eksekusi standar dan kustom memberi Anda video dan log untuk pengujian Anda. Namun, dalam mode standar, Anda mendapatkan video dan log pengujian yang telah ditentukan sebelumnya hanya setelah pengujian Anda selesai.

   Sebaliknya, mode kustom memberi Anda streaming langsung video dan log sisi klien pengujian Anda. Selain itu, Anda dapat mengunduh video dan artefak lainnya di akhir tes.

4. Pengakhiran: Jenis pengujian berikut akan dihentikan pada akhir Desember 2023 dalam mode eksekusi standar:

   - Appium (semua bahasa)
   - Labu
   - XCTest
   - Otomatisasi UI
   - Automator UI
   - Tes Web
   - Penjelajah bawaan

   Setelah usang, Anda tidak akan dapat menggunakan kerangka kerja ini dalam mode standar. Sebagai gantinya, Anda dapat menggunakan mode khusus untuk jenis pengujian yang tercantum di atas.

> **Tip**
>
> Jika kasus penggunaan Anda melibatkan setidaknya satu dari faktor di atas, kami sangat menyarankan untuk beralih ke mode eksekusi kustom.

Langkah migrasi

Untuk bermigrasi dari mode standar ke mode khusus, lakukan hal berikut:

1. Masuk ke AWS Management Console dan buka konsol Device Farm di https://console.aws.amazon.com/devicefarm/.
2. Pilih proyek Anda dan kemudian mulai menjalankan otomatisasi baru.

3. Unggah aplikasi Anda (atau pilih`web app`), pilih jenis kerangka pengujian Anda, unggah paket pengujian Anda, lalu di bawah `Choose your execution environment` parameter, pilih opsi untuk`Run your test in a custom environment`.
4. Secara default, file spesifikasi pengujian contoh Device Farm akan muncul untuk Anda lihat dan edit. File contoh ini dapat digunakan sebagai tempat awal untuk mencoba pengujian Anda dalam mode lingkungan khusus. Kemudian, setelah Anda memverifikasi bahwa pengujian Anda berfungsi dengan baik dari konsol, Anda kemudian dapat mengubah integrasi API, CLI, dan pipeline apa pun dengan Device Farm untuk menggunakan file spesifikasi pengujian ini sebagai parameter saat penjadwalan pengujian berjalan. Untuk informasi tentang cara menambahkan file spesifikasi pengujian sebagai parameter untuk proses Anda, lihat bagian `testSpecArn` parameter untuk `ScheduleRun` API di panduan API kami.

Kerangka Appium

Dalam lingkungan pengujian khusus, Device Farm tidak menyisipkan atau mengganti kemampuan Appium apa pun dalam pengujian kerangka kerja Appium Anda. Anda harus menentukan kemampuan Appium pengujian Anda baik dalam file YAMM spesifikasi pengujian atau kode pengujian Anda.

Instrumentasi Android

Anda tidak perlu membuat perubahan untuk memindahkan pengujian instrumentasi Android ke lingkungan pengujian khusus.

iOS XCuitest

Anda tidak perlu membuat perubahan untuk memindahkan pengujian XCUITest iOS Anda ke lingkungan pengujian khusus.

Memperluas lingkungan pengujian khusus di Device Farm

Mode Kustom Device Farm memungkinkan Anda menjalankan lebih dari sekadar rangkaian pengujian. Di bagian ini, Anda mempelajari cara memperluas rangkaian pengujian dan mengoptimalkan pengujian Anda.

Mengatur PIN

Beberapa aplikasi mengharuskan Anda mengatur PIN pada perangkat. Device Farm tidak mendukung pengaturan PIN pada perangkat secara native. Namun, ini dimungkinkan dengan peringatan berikut:

- Perangkat harus menjalankan Android 8 atau lebih tinggi.
- PIN harus dilepas setelah tes selesai.

Untuk mengatur PIN dalam pengujian Anda, gunakan `post_test` fase `pre_test` dan untuk mengatur dan menghapus PIN, seperti yang ditunjukkan berikut:

```
phases:
    pre_test:
      - # ... among your pre_test commands
      - DEVICE_PIN_CODE="1234"
      - adb shell locksettings set-pin "$DEVICE_PIN_CODE"
    post_test:
      - # ... Among your post_test commands
      - adb shell locksettings clear --old "$DEVICE_PIN_CODE"
```

Saat rangkaian pengujian Anda dimulai, PIN 1234 disetel. Setelah rangkaian pengujian Anda keluar, PIN akan dihapus.

> **Warning**
>
> Jika Anda tidak menghapus PIN dari perangkat setelah pengujian selesai, perangkat dan akun Anda akan dikarantina.

Mempercepat tes berbasis Appium melalui kemampuan yang diinginkan

Saat menggunakan Appium, Anda mungkin menemukan bahwa rangkaian pengujian mode standar sangat lambat. Ini karena Device Farm menerapkan pengaturan default dan tidak membuat asumsi tentang bagaimana Anda ingin menggunakan lingkungan Appium. Meskipun default ini dibangun di sekitar praktik terbaik industri, mereka mungkin tidak berlaku untuk situasi Anda. Untuk

menyempurnakan parameter server Appium, Anda dapat menyesuaikan kemampuan Appium default dalam spesifikasi pengujian Anda. Misalnya, berikut ini menyetel `usePrebuildWDA` kemampuan ke `true` dalam rangkaian pengujian iOS untuk mempercepat waktu mulai awal:

```
phases:
  pre_test:
    - # ... Start up Appium
    - >-
    appium --log-timestamp
    --default-capabilities "{\"usePrebuiltWDA\": true, \"derivedDataPath\":
\"$DEVICEFARM_WDA_DERIVED_DATA_PATH\",
    \"deviceName\": \"$DEVICEFARM_DEVICE_NAME\", \"platformName\":
\"$DEVICEFARM_DEVICE_PLATFORM_NAME\", \"app\":\"$DEVICEFARM_APP_PATH\",
    \"automationName\":\"XCUITest\", \"udid\":\"$DEVICEFARM_DEVICE_UDID_FOR_APPIUM\",
 \"platformVersion\":\"$DEVICEFARM_DEVICE_OS_VERSION\"}"
    >> $DEVICEFARM_LOG_DIR/appiumlog.txt 2>&1 &
```

Kemampuan Appium harus berupa struktur JSON yang dikutip dari cangkang.

Kemampuan Appium berikut adalah sumber umum peningkatan kinerja:

`noReset` dan `fullReset`

Kedua kemampuan ini, yang saling eksklusif, menggambarkan perilaku Appium setelah setiap sesi selesai. Ketika `noReset` disetel `ketrue`, server Appium tidak menghapus data dari aplikasi Anda ketika sesi Appium berakhir, secara efektif tidak melakukan pembersihan apa pun. `fullReset`menghapus instalasi dan menghapus semua data aplikasi dari perangkat setelah sesi ditutup. Untuk informasi selengkapnya, lihat Reset Strategi dalam dokumentasi Appium.

`ignoreUnimportantViews`(Hanya Android)

Menginstruksikan Appium untuk mengompres hierarki UI Android hanya ke tampilan yang relevan untuk pengujian, mempercepat pencarian elemen tertentu. Namun, ini dapat merusak beberapa rangkaian pengujian berbasis XPath karena hierarki tata letak UI telah diubah.

`skipUnlock`(Hanya Android)

Menginformasikan Appium bahwa tidak ada kode PIN yang saat ini disetel, yang mempercepat pengujian setelah peristiwa layar mati atau peristiwa kunci lainnya.

`webDriverAgentUrl`(Hanya iOS)

Menginstruksikan Appium untuk menganggap bahwa ketergantungan iOS penting,`webDriverAgent`, sudah berjalan dan tersedia untuk menerima permintaan HTTP di URL yang ditentukan. Jika `webDriverAgent` belum aktif dan berjalan, Appium membutuhkan waktu beberapa saat di awal rangkaian pengujian untuk memulai. `webDriverAgent` Jika Anda memulai `webDriverAgent` sendiri dan mengatur `webDriverAgentUrl` ke `http://localhost:8100` saat memulai Appium, Anda dapat mem-boot suite pengujian Anda lebih cepat. Perhatikan bahwa kemampuan ini tidak boleh digunakan bersama `useNewWDA` kemampuan.

Anda dapat menggunakan kode berikut untuk memulai `webDriverAgent` dari file spesifikasi pengujian di port lokal perangkat`8100`, lalu meneruskannya ke port lokal host pengujian `8100` (ini memungkinkan Anda untuk menetapkan `webDriverAgentUrl` nilainyah`http://localhost:8100`). Kode ini harus dijalankan selama fase penginstalan setelah kode apa pun untuk menyiapkan variabel Appium dan `webDriverAgent` lingkungan telah ditentukan:

```
      # Start WebDriverAgent and iProxy
      - >-
        xcodebuild test-without-building -project /usr/local/avm/versions/
$APPIUM_VERSION/node_modules/appium/node_modules/appium-webdriveragent/
WebDriverAgent.xcodeproj
        -scheme WebDriverAgentRunner -derivedDataPath
 $DEVICEFARM_WDA_DERIVED_DATA_PATH
        -destination id=$DEVICEFARM_DEVICE_UDID_FOR_APPIUM
 IPHONEOS_DEPLOYMENT_TARGET=$DEVICEFARM_DEVICE_OS_VERSION
        GCC_TREAT_WARNINGS_AS_ERRORS=0 COMPILER_INDEX_STORE_ENABLE=NO >>
 $DEVICEFARM_LOG_DIR/webdriveragent_log.txt 2>&1 &

        iproxy 8100 8100 >> $DEVICEFARM_LOG_DIR/iproxy_log.txt 2>&1 &
```

Kemudian, Anda dapat menambahkan kode berikut ke file spesifikasi pengujian Anda untuk memastikannya berhasil `webDriverAgent` dimulai. Kode ini harus dijalankan pada akhir fase pra-pengujian setelah memastikan bahwa Appium berhasil dimulai:

```
      # Wait for WebDriverAgent to start
      - >-
        start_wda_timeout=0;
        while [ true ];
        do
```

```
          if [ $start_wda_timeout -gt 60 ];
          then
              echo "WebDriverAgent server never started in 60 seconds.";
              exit 1;
          fi;
          grep -i "ServerURLHere" $DEVICEFARM_LOG_DIR/webdriveragent_log.txt >> /
dev/null 2>&1;
          if [ $? -eq 0 ];
          then
              echo "WebDriverAgent REST http interface listener started";
              break;
          else
              echo "Waiting for WebDriverAgent server to start. Sleeping for 1
 seconds";
              sleep 1;
              start_wda_timeout=$((start_wda_timeout+1));
          fi;
        done;
```

Untuk informasi selengkapnya tentang kemampuan yang didukung Appium, lihat Kemampuan yang Diinginkan Appium dalam dokumentasi Appium.

Menggunakan Webhook dan API lainnya setelah pengujian Anda dijalankan

Anda dapat meminta Device Farm memanggil webhook setelah setiap rangkaian pengujian selesai digunakan. curl Proses untuk melakukan ini bervariasi dengan tujuan dan pemformatan. Untuk webhook spesifik Anda, lihat dokumentasi untuk webhook tersebut. Contoh berikut memposting pesan setiap kali rangkaian pengujian selesai ke webhook Slack:

```
phases:
  post_test:
    - curl -X POST -H 'Content-type: application/json' --data '{"text":"Tests on
 '$DEVICEFARM_DEVICE_NAME' have finished!"}' https://hooks.slack.com/services/
T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX
```

Untuk informasi selengkapnya tentang penggunaan webhook dengan Slack, lihat Mengirim pesan Slack pertama Anda menggunakan Webhook di referensi Slack API.

Anda tidak terbatas pada menggunakan curl untuk memanggil webhooks. Paket pengujian dapat menyertakan skrip dan alat tambahan, asalkan kompatibel dengan lingkungan eksekusi Device Farm.

Misalnya, paket pengujian Anda mungkin menyertakan skrip tambahan yang membuat permintaan ke API lain. Pastikan bahwa setiap paket yang diperlukan diinstal bersamaan dengan persyaratan suite pengujian Anda. Untuk menambahkan skrip yang berjalan setelah rangkaian pengujian Anda selesai, sertakan skrip dalam paket pengujian Anda dan tambahkan yang berikut ini ke spesifikasi pengujian Anda:

```
phases:
  post_test:
    - python post_test.py
```

> **Note**
>
> Mempertahankan kunci API atau token otentikasi lain yang digunakan dalam paket pengujian Anda adalah tanggung jawab Anda. Kami menyarankan agar Anda menyimpan segala bentuk kredensi keamanan di luar kendali sumber, menggunakan kredensil dengan hak istimewa sesedikit mungkin, dan menggunakan token yang dapat diubah dan berumur pendek bila memungkinkan. Untuk memverifikasi persyaratan keamanan, lihat dokumentasi untuk API pihak ketiga yang Anda gunakan.

Jika Anda berencana menggunakan AWS layanan sebagai bagian dari rangkaian eksekusi pengujian, Anda harus menggunakan kredenal sementara IAM, yang dihasilkan di luar rangkaian pengujian dan disertakan dalam paket pengujian Anda. Kredensi ini harus memiliki izin paling sedikit yang diberikan dan umur sesingkat mungkin. Untuk informasi selengkapnya tentang membuat kredensil sementara, lihat Meminta kredensil keamanan sementara di Panduan Pengguna IAM.

Menambahkan file tambahan ke paket pengujian Anda

Anda mungkin ingin menggunakan file tambahan sebagai bagian dari pengujian Anda baik sebagai file konfigurasi tambahan atau data pengujian tambahan. Anda dapat menambahkan file tambahan ini ke paket pengujian sebelum mengunggahnyaAWS Device Farm, lalu mengaksesnya dari mode lingkungan khusus. Pada dasarnya, semua format unggahan paket uji (ZIP, IPA, APK, JAR, dll.) Adalah format arsip paket yang mendukung operasi ZIP standar.

Anda dapat menambahkan file ke arsip pengujian sebelum mengunggahnya AWS Device Farm dengan menggunakan perintah berikut:

```
$ zip zip-with-dependencies.zip extra_file
```

Untuk direktori file tambahan:

```
$ zip -r zip-with-dependencies.zip extra_files/
```

Perintah ini berfungsi seperti yang diharapkan untuk semua format unggahan paket pengujian kecuali untuk file IPA. Untuk file IPA, terutama saat digunakan dengan XCUITests, kami sarankan Anda meletakkan file tambahan di lokasi yang sedikit berbeda karena cara mengundurkan diri AWS Device Farm paket uji iOS. Saat membuat pengujian iOS Anda, direktori aplikasi pengujian akan berada di dalam direktori lain bernama *`Payload`*.

Misalnya, ini adalah bagaimana satu direktori pengujian iOS seperti itu terlihat:

```
$ tree
.
### Payload
    ### ADDiOSReferenceAppUITests-Runner.app
        ### ADDiOSReferenceAppUITests-Runner
        ### Frameworks
        #   ### XCTAutomationSupport.framework
        #   #   ### Info.plist
        #   #   ### XCTAutomationSupport
        #   #   ### _CodeSignature
        #   #   #   ### CodeResources
        #   #   ### version.plist
        #   ### XCTest.framework
        #       ### Info.plist
        #       ### XCTest
        #       ### _CodeSignature
        #       #   ### CodeResources
        #       ### en.lproj
        #       #   ### InfoPlist.strings
        #       ### version.plist
        ### Info.plist
        ### PkgInfo
        ### PlugIns
        #   ### ADDiOSReferenceAppUITests.xctest
        #   #   ### ADDiOSReferenceAppUITests
        #   #   ### Info.plist
        #   #   ### _CodeSignature
        #   #       ### CodeResources
        #   ### ADDiOSReferenceAppUITests.xctest.dSYM
        #       ### Contents
```

```
        #           ### Info.plist
        #           ### Resources
        #               ### DWARF
        #                   ### ADFiOSReferenceAppUITests
        ### _CodeSignature
        #   ### CodeResources
        ### embedded.mobileprovision
```

Untuk paket XCUITest ini, tambahkan file tambahan apa pun ke direktori yang diakhiri dengan.app di dalam direktori Payload. Misalnya, perintah berikut menunjukkan bagaimana Anda dapat menambahkan file ke paket pengujian ini:

```
$ mv extra_file Payload/*.app/
$ zip -r my_xcui_tests.ipa Payload/
```

Saat menambahkan file ke paket pengujian, Anda dapat mengharapkan perilaku interaksi yang sedikit berbeda AWS Device Farm berdasarkan format unggahannya. Jika unggahan menggunakan ekstensi file ZIP, AWS Device Farm akan secara otomatis membuka zip unggahan sebelum pengujian Anda dan meninggalkan file yang tidak di-zip di lokasi dengan variabel lingkungan *$DEVICEFARM_TEST_PACKAGE_PATH*. (Ini berarti bahwa jika Anda menambahkan file bernama *extra_file* ke root arsip seperti pada contoh pertama, itu akan terletak di *$deviceFarm_test_package_path/extra_file* selama pengujian).

Untuk menggunakan contoh yang lebih praktis, jika Anda adalah pengguna Appium TestNG yang ingin menyertakan file *testng.xml* dengan pengujian Anda, Anda dapat memasukkannya ke dalam arsip menggunakan perintah berikut:

```
$ zip zip-with-dependencies.zip testng.xml
```

Kemudian, Anda dapat mengubah perintah pengujian Anda dalam mode lingkungan khusus menjadi berikut:

```
java -D appium.screenshots.dir=$DEVICEFARM_SCREENSHOT_PATH org.testng.TestNG -testjar
 *-tests.jar -d $DEVICEFARM_LOG_DIR/test-output $DEVICEFARM_TEST_PACKAGE_PATH/
testng.xml
```

Jika ekstensi unggahan paket pengujian Anda bukan ZIP (misalnya, file APK, IPA, atau JAR), file paket yang diunggah itu sendiri ditemukan di $DEVICEFARM_TEST_PACKAGE_PATH. Karena ini masih file format arsip, Anda dapat unzip file

untuk mengakses file tambahan dari dalam. Misalnya, perintah berikut akan membuka zip isi paket pengujian (untuk file APK, IPA, atau JAR) ke direktori */tmp*:

```
unzip $DEVICEFARM_TEST_PACKAGE_PATH -d /tmp
```

Dalam kasus file APK atau JAR, Anda akan menemukan file tambahan Anda di-unzip ke direktori /tmp (mis., /tmp/extra_file). Dalam kasus file IPA, seperti yang dijelaskan sebelumnya, file tambahan akan berada di lokasi yang sedikit berbeda di dalam folder yang diakhiri *dengan.app*, yang berada di dalam direktori *Payload*. *Misalnya, berdasarkan contoh IPA di atas, file akan ditemukan di lokasi /tmp/payload/AdFiOS UITests-runner.app/extra_file (dapat direferensikan sebagai /tmp/payload/ ReferenceApp *.app/extra_file).*

Bekerja dengan akses jarak jauh di AWS Device Farm

Akses jarak jauh memungkinkan Anda untuk menggesek, memberi isyarat, dan berinteraksi dengan perangkat melalui browser web Anda secara real time untuk menguji fungsionalitas dan mereproduksi masalah pelanggan. Anda berinteraksi dengan perangkat tertentu dengan membuat sesi akses jarak jauh dengan perangkat tersebut.

Sesi di Device Farm adalah interaksi real-time dengan perangkat fisik aktual yang dihosting di browser web. Sesi menampilkan perangkat tunggal yang Anda pilih saat memulai sesi. Seorang pengguna dapat memulai lebih dari satu sesi pada satu waktu dengan jumlah total perangkat simultan dibatasi oleh jumlah slot perangkat yang Anda miliki. Anda dapat membeli slot perangkat berdasarkan keluarga perangkat (perangkat Android atau iOS). Untuk informasi lebih lanjut, lihatHarga Device Farm.

Device Farm saat ini menawarkan subset perangkat untuk pengujian akses jarak jauh. Perangkat baru ditambahkan ke kumpulan perangkat setiap saat.

Device Farm menangkap video dari setiap sesi akses jarak jauh dan menghasilkan log aktivitas selama sesi berlangsung. Hasil ini mencakup informasi apa pun yang Anda berikan selama sesi.

> **Note**
>
> Untuk alasan keamanan, kami menyarankan Anda menghindari memberikan atau memasukkan informasi sensitif, seperti nomor akun, informasi login pribadi, dan detail lainnya selama sesi akses jarak jauh.

Topik

Membuat sesi akses jarak jauh di AWS Device Farm

Untuk informasi tentang sesi akses jarak jauh, lihatSesi.

Prasyarat

- Buat proyek di Device Farm. Ikuti instruksi diMembuat proyek di AWS Device Farm, dan kemudian kembali ke halaman ini.

Buat sesi dengan konsol Device Farm

1. Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.
3. Jika Anda sudah memiliki proyek, pilih dari daftar. Jika tidak, buat proyek dengan mengikuti instruksi diMembuat proyek di AWS Device Farm.
4. PadaAkses jarak jauhtab, pilihMemulai sesi baru.
5. Pilih perangkat untuk sesi Anda. Anda dapat memilih dari daftar perangkat yang tersedia atau mencari perangkat menggunakan bilah pencarian di bagian atas daftar. Anda dapat mencari berdasarkan:
   - Nama
   - Platform
   - Faktor bentuk
   - Jenis armada
6. DiNama sesi, masukkan nama untuk sesi tersebut.
7. PilihKonfirmasikan dan mulai sesi.

Langkah selanjutnya

Device Farm memulai sesi segera setelah perangkat yang diminta tersedia, biasanya dalam beberapa menit. ThePerangkat Dimintakotak dialog muncul sampai sesi dimulai. Untuk membatalkan permintaan sesi, pilihBatalkan permintaan.

Setelah sesi dimulai, jika Anda harus menutup browser atau tab browser tanpa menghentikan sesi atau jika koneksi antara browser dan internet terputus, sesi tetap aktif selama lima menit. Setelah itu, Device Farm mengakhiri sesi. Akun Anda dikenakan biaya untuk waktu idle.

Setelah sesi dimulai, Anda dapat berinteraksi dengan perangkat di browser web.

Menggunakan sesi akses jarak jauh di AWS Device Farm

Untuk informasi tentang melakukan pengujian interaktif aplikasi Android dan iOS melalui sesi akses jarak jauh, lihatSesi.

- Prasyarat
- Menggunakan sesi di konsol Device Farm
- Langkah selanjutnya
- Kiat dan trik

Prasyarat

- Buat sesi. Ikuti instruksi diBuat sesi, dan kemudian kembali ke halaman ini.

Menggunakan sesi di konsol Device Farm

Segera setelah perangkat yang Anda minta untuk sesi akses jarak jauh tersedia, konsol akan menampilkan layar perangkat. Sesi ini memiliki panjang maksimum 150 menit. Waktu yang tersisa dalam sesi muncul diWaktu Tersisabidang dekat nama perangkat.

Menginstal aplikasi

Untuk menginstal aplikasi pada perangkat sesi, diInstal aplikasi, pilihPilih File, lalu pilih file.apk (Android) atau file.ipa (iOS) yang ingin Anda instal. Aplikasi yang Anda jalankan dalam sesi akses jarak jauh tidak memerlukan instrumentasi pengujian atau penyediaan apa pun.

> **Note**
>
> AWS Device Farm tidak menampilkan konfirmasi setelah aplikasi diinstal. Coba berinteraksi dengan ikon aplikasi untuk melihat apakah aplikasi siap digunakan.

Saat Anda mengunggah aplikasi, terkadang ada penundaan sebelum aplikasi tersedia. Lihat baki sistem untuk menentukan apakah aplikasi tersedia.

Mengontrol perangkat

Anda dapat berinteraksi dengan perangkat yang ditampilkan di konsol seperti halnya perangkat fisik yang sebenarnya, dengan menggunakan mouse Anda atau perangkat yang sebanding untuk sentuhan dan keyboard di layar perangkat. Untuk perangkat Android, ada tombol diLihat kontrolFungsi tersebut sepertiRumahdanKembalitombol pada perangkat Android. Untuk perangkat iOS, adaRumahtombol yang berfungsi seperti tombol beranda pada perangkat iOS. Anda juga dapat beralih di antara aplikasi yang berjalan di perangkat dengan memilihAplikasi Terbaru.

Beralih antara mode potret dan lanskap

Anda juga dapat beralih antara mode potret (vertikal) dan lanskap (horizontal) untuk perangkat yang Anda gunakan.

Langkah selanjutnya

Device Farm melanjutkan sesi hingga Anda menghentikannya secara manual atau batas waktu 150 menit tercapai. Untuk mengakhiri sesi, pilihHentikan Sesi. Setelah sesi berhenti, Anda dapat mengakses video yang diambil dan log yang dihasilkan. Untuk informasi selengkapnya, lihat Dapatkan hasil sesi.

Kiat dan trik

Anda mungkin mengalami masalah kinerja dengan sesi akses jarak jauh di beberapaAWSDaerah. Ini sebagian disebabkan oleh latensi di beberapa Wilayah. Jika Anda mengalami masalah kinerja, berikan sesi akses jarak jauh kesempatan untuk mengejar ketinggalan sebelum Anda berinteraksi dengan aplikasi lagi.

Dapatkan hasil sesi akses jarak jauh di AWS Device Farm

Untuk informasi tentang sesi, lihatSesi.

- Prasyarat
- Melihat detail sesi

Prasyarat

- Selesaikan sesi. Ikuti instruksi diMenggunakan sesi akses jarak jauh di AWS Device Farm, dan kemudian kembali ke halaman ini.

Melihat detail sesi

Saat sesi akses jarak jauh berakhir, konsol Device Farm menampilkan tabel yang berisi detail tentang aktivitas selama sesi berlangsung. Untuk informasi lebih lanjut, lihatMenganalisis Informasi Log.

Untuk kembali ke detail sesi di lain waktu:

1. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.
2. Pilih proyek yang berisi sesi.
3. PilihAkses jarak jauh, lalu pilih sesi yang ingin Anda tinjau dari daftar.

Mengunduh video sesi atau log

Saat sesi akses jarak jauh berakhir, konsol Device Farm menyediakan akses ke rekaman video sesi dan log aktivitas. Dalam hasil sesi, pilihBerkastab untuk daftar link ke video sesi dan log. Anda dapat melihat file-file ini di browser atau menyimpannya secara lokal.

Bekerja dengan perangkat pribadi di AWS Device Farm

Perangkat pribadi adalah perangkat seluler fisik yang digunakan AWS Device Farm atas nama Anda di pusat data Amazon. Perangkat ini eksklusif untuk AWS akun Anda.

> **Note**
>
> Saat ini, perangkat pribadi hanya tersedia di Wilayah AWS AS Barat (Oregon) (`us-west-2`).

Jika Anda memiliki armada perangkat pribadi, Anda dapat membuat sesi akses jarak jauh dan menjadwalkan pengujian berjalan dengan perangkat pribadi Anda. Anda juga dapat membuat profil instance untuk mengontrol perilaku perangkat pribadi Anda selama sesi akses jarak jauh atau uji coba. Untuk informasi selengkapnya, lihat Mengelola perangkat pribadi di AWS Device Farm. Secara opsional, Anda dapat meminta agar perangkat pribadi Android tertentu digunakan sebagai perangkat yang di-rooting.

Anda juga dapat membuat layanan endpoint Amazon Virtual Private Cloud untuk menguji aplikasi pribadi yang dapat diakses perusahaan Anda, tetapi tidak dapat dijangkau melalui internet. Misalnya, Anda mungkin memiliki aplikasi web yang berjalan di VPC yang ingin Anda uji di perangkat seluler. Untuk informasi selengkapnya, lihat Menggunakan layanan endpoint Amazon VPC dengan Device Farm.

Jika Anda tertarik untuk menggunakan armada perangkat pribadi, hubungi kami. Tim Device Farm harus bekerja sama dengan Anda untuk menyiapkan dan menyebarkan armada perangkat pribadi untuk AWS akun Anda.

Topik

Mengelola perangkat pribadi di AWS Device Farm

Perangkat pribadi adalah perangkat seluler fisik yang digunakan AWS Device Farm atas nama Anda di pusat data Amazon. Perangkat ini eksklusif untuk AndaAWSakun.

> **Note**
>
> Saat ini, perangkat pribadi tersedia diAWSWilayah AS Barat (Oregon) (`us-west-2`) hanya.

Anda dapat mengatur armada yang berisi satu atau lebih perangkat pribadi. Perangkat ini didedikasikan untukAWSakun. Setelah menyiapkan perangkat, Anda dapat membuat satu atau beberapa profil instans secara opsional untuk perangkat tersebut. Profil instans dapat membantu Anda mengotomatiskan proses pengujian dan secara konsisten menerapkan pengaturan yang sama ke instance perangkat.

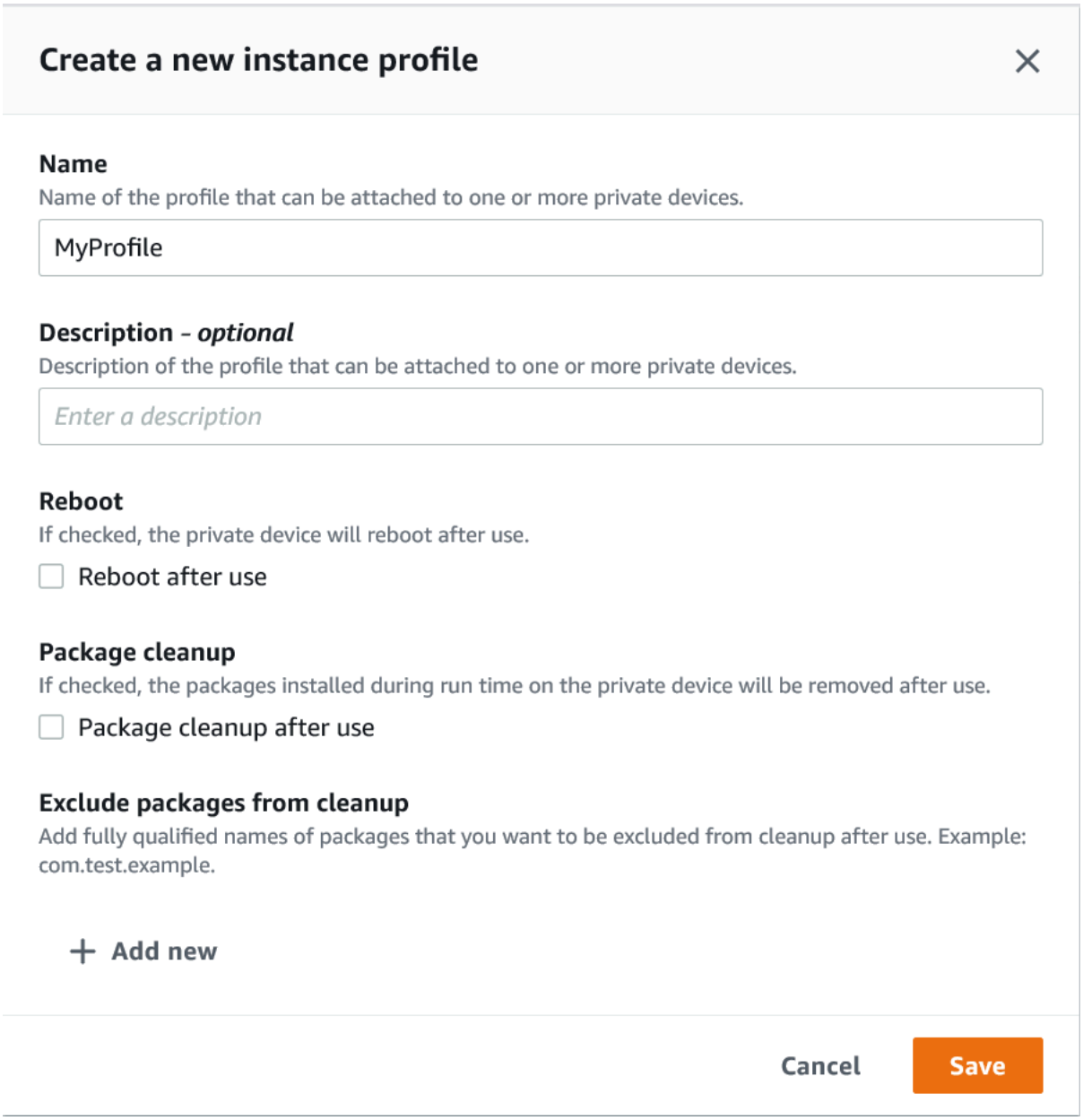
Topik ini menjelaskan cara membuat profil instance dan melakukan tugas manajemen perangkat umum lainnya.

Topik

Membuat profil instans

Untuk mengontrol perilaku perangkat pribadi selama sesi uji coba atau akses jarak jauh, Anda dapat membuat atau memodifikasi profil instance di Device Farm. Anda tidak memerlukan profil instans untuk mulai menggunakan perangkat pribadi Anda.

1. Buka konsol Device Farm dihttps://console.aws.amazon.com/devicefarm/.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihPerangkat pribadi.
3. PilihProfil instans.
4. PilihBuat profil contoh.
5. Masukkan nama untuk profil instance.



6. (Opsional) Masukkan deskripsi untuk profil instance.
7. (Opsional) Ubah salah satu pengaturan berikut untuk menentukan tindakan yang Anda inginkan Device Farm lakukan pada perangkat setelah setiap pengujian atau sesi berakhir:

   - Reboot setelah digunakan— Untuk me-reboot perangkat, pilih kotak centang ini. Secara default, kotak centang ini dihapus (`false`).
   - Pembersihan paket— Untuk menghapus semua paket aplikasi yang Anda instal pada perangkat, pilih kotak centang ini. Secara default, kotak centang ini dihapus (`false`). Untuk menyimpan semua paket aplikasi yang Anda instal di perangkat, biarkan kotak centang ini dihapus.

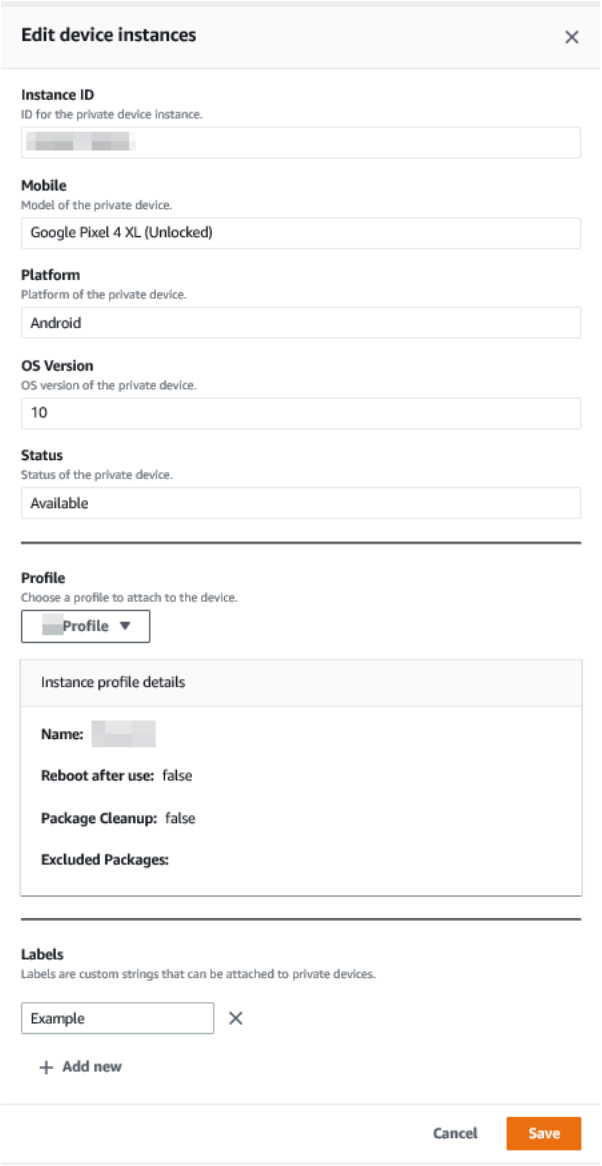
- Kecualikan paket dari pembersihan— Untuk menyimpan hanya paket aplikasi yang dipilih di perangkat, pilihPembersihan Paketcentang kotak, lalu pilihTambahkan baru. Untuk nama paket, masukkan nama paket aplikasi yang sepenuhnya memenuhi syarat yang ingin Anda simpan di perangkat (misalnya,`com.test.example`). Untuk menyimpan lebih banyak paket aplikasi di perangkat, pilihTambahkan baru, dan kemudian masukkan nama yang sepenuhnya memenuhi syarat dari setiap paket.

8. Pilih Save (Simpan).

Mengelola instance perangkat pribadi

Jika Anda sudah memiliki satu atau beberapa perangkat pribadi di armada, Anda dapat melihat informasi tentang dan mengelola pengaturan tertentu untuk setiap instance perangkat. Anda juga dapat meminta instance perangkat pribadi tambahan.

1. Buka konsol Device Farm dihttps://console.aws.amazon.com/devicefarm/.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihPerangkat pribadi.
3. PilihInstans perangkat. TheInstans perangkattab menampilkan tabel perangkat pribadi yang ada di armada Anda. Untuk mencari atau memfilter tabel dengan cepat, masukkan istilah pencarian di bilah pencarian di atas kolom.
4. (Opsional) Untuk meminta instance perangkat pribadi baru, pilihMinta contoh perangkatatauhubungi kami. Perangkat pribadi memerlukan pengaturan tambahan dengan bantuan dari tim Device Farm.
5. Dalam tabel instance perangkat, pilih opsi sakelar di sebelah instance yang ingin Anda lihat atau kelola informasi, lalu pilihSunting.



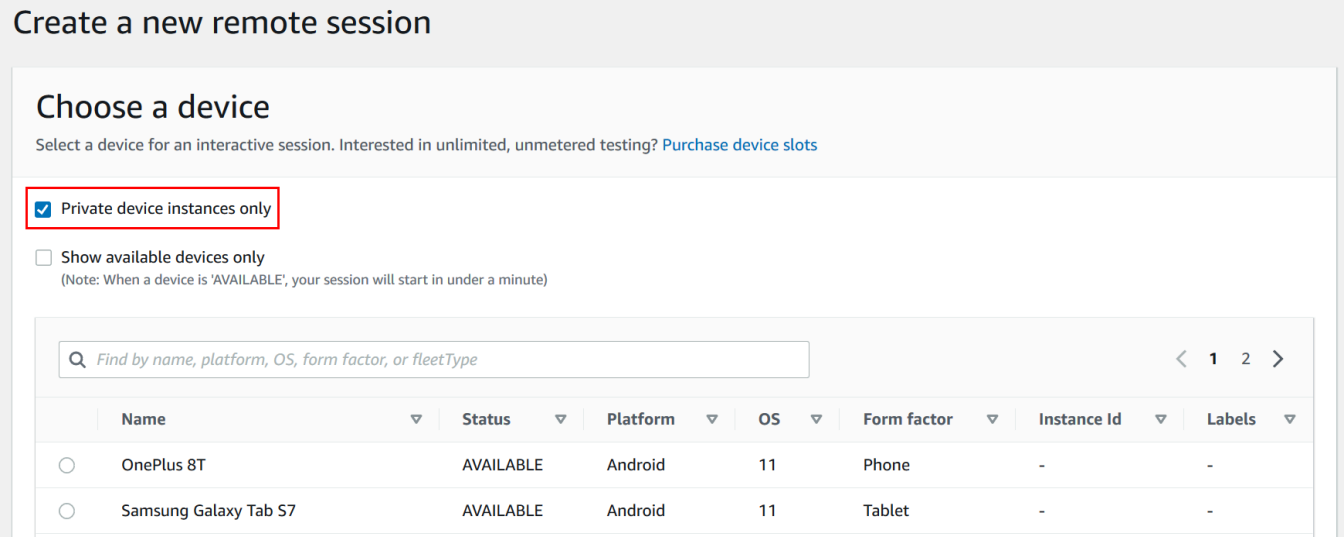
6. (Opsional) UntukProfil, pilih profil instance untuk dilampirkan ke instance perangkat. Ini dapat membantu jika Anda ingin selalu mengecualikan paket aplikasi tertentu dari tugas pembersihan, misalnya.
7. (Opsional) Di bawahLabel, pilihTambahkan baruuntuk menambahkan label ke instance perangkat. Label dapat membantu Anda mengkategorikan perangkat Anda dan menemukan perangkat tertentu dengan lebih mudah.
8. Pilih Save (Simpan).

Membuat uji coba atau memulai sesi akses jarak jauh

Setelah menyiapkan armada perangkat pribadi, Anda dapat membuat uji coba atau memulai sesi akses jarak jauh dengan satu atau beberapa perangkat pribadi di armada Anda.

1. Buka konsol Device Farm dihttps://console.aws.amazon.com/devicefarm/.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.
3. Pilih proyek yang sudah ada dari daftar atau buat yang baru. Untuk membuat proyek baru, pilihProyek baru, masukkan nama untuk proyek, lalu pilihKirim.
4. Lakukan salah satu dari berikut:

   - Untuk membuat uji coba, pilihTes otomatis, dan kemudian pilihBuat run baru. Wizard memandu Anda melalui langkah-langkah untuk membuat run. UntukPilih perangkatlangkah, Anda dapat mengedit kumpulan perangkat yang ada atau membuat kumpulan perangkat baru yang hanya menyertakan perangkat pribadi yang disiapkan dan diasosiasikan oleh tim Device FarmAWSakun. Untuk informasi selengkapnya, lihat the section called “Buat kolam perangkat pribadi”.
   - Untuk memulai sesi akses jarak jauh, pilihAkses jarak jauh, dan kemudian pilihMemulai sesi baru. PadaPilih perangkathalaman, pilihInstans perangkat pribadi sajauntuk membatasi daftar hanya pada perangkat pribadi yang disiapkan dan dikaitkan dengan tim Device FarmAWSakun. Kemudian, pilih perangkat yang ingin Anda akses, masukkan nama untuk sesi akses jarak jauh, dan pilihKonfirmasikan dan mulai sesi.



Langkah selanjutnya

Setelah menyiapkan perangkat pribadi, Anda juga dapat mengelola perangkat pribadi dengan cara berikut:

- Lewati penandatanganan ulang aplikasi di perangkat pribadi
- Menggunakan layanan endpoint Amazon Virtual Private Cloud dengan Device Farm

Untuk menghapus profil instance, padaProfil instansmenu, pilih opsi sakelar di sebelah instance yang ingin Anda hapus, lalu pilihHapus.

Memilih perangkat pribadi di kumpulan perangkat

Untuk menggunakan perangkat pribadi dalam uji coba, Anda dapat membuat kumpulan perangkat yang memilih perangkat pribadi Anda. Kumpulan perangkat memungkinkan Anda memilih perangkat pribadi terutama melalui tiga jenis aturan kumpulan perangkat:

1. Aturan berdasarkan perangkat ARN
2. Aturan berdasarkan label instance perangkat
3. Aturan berdasarkan contoh perangkat ARN

Pada bagian berikut, setiap jenis aturan dan kasus penggunaannya dijelaskan secara mendalam. Anda dapat menggunakan konsol Device Farm,AWSAntarmuka Baris Perintah (AWSCLI), atau Device Farm API untuk membuat atau memodifikasi kumpulan perangkat dengan perangkat pribadi menggunakan aturan ini.

Topik

- Perangkat ARN
- Label instance perangkat
- Contoh ARN
- Membuat kolam perangkat pribadi dengan perangkat pribadi (konsol)
- Membuat kolam perangkat pribadi dengan perangkat pribadi (AWS CLI)
- Membuat kumpulan perangkat pribadi dengan perangkat pribadi (API)

Perangkat ARN

Perangkat ARN adalah pengidentifikasi yang mewakili jenis perangkat daripada instance perangkat fisik tertentu. Jenis perangkat ditentukan oleh atribut berikut:

- ID armada perangkat
- OEM perangkat
- Nomor model perangkat
- Versi sistem operasi perangkat
- Status perangkat yang menunjukkan apakah itu di-root atau tidak

Banyak instance perangkat fisik dapat diwakili oleh satu jenis perangkat di mana setiap instance dari tipe tersebut memiliki nilai yang sama untuk atribut ini. Misalnya, jika Anda memiliki tiga`Apple iPhone 13`perangkat pada versi iOS`16.1.0`di armada pribadi Anda, setiap perangkat akan berbagi perangkat ARN yang sama. Jika ada perangkat yang ditambahkan atau dihapus dari armada Anda dengan atribut yang sama, perangkat ARN akan terus mewakili perangkat apa pun yang tersedia yang Anda miliki di armada Anda untuk jenis perangkat tersebut.

Perangkat ARN adalah cara paling kuat untuk memilih perangkat pribadi untuk kumpulan perangkat karena memungkinkan kumpulan perangkat untuk terus memilih perangkat terlepas dari instance perangkat tertentu yang telah Anda gunakan pada waktu tertentu. Instance perangkat pribadi individu dapat mengalami kegagalan perangkat keras, mendorong Device Farm untuk secara otomatis menggantinya dengan instance kerja baru dari jenis perangkat yang sama. Dalam skenario ini, aturan ARN perangkat memastikan bahwa kumpulan perangkat Anda dapat terus memilih perangkat jika terjadi kegagalan perangkat keras.

Saat Anda menggunakan aturan ARN perangkat untuk perangkat pribadi di kumpulan perangkat Anda dan menjadwalkan pengujian yang dijalankan dengan kumpulan tersebut, Device Farm akan secara otomatis memeriksa instance perangkat pribadi mana yang diwakili oleh ARN perangkat tersebut. Dari contoh yang saat ini tersedia, salah satunya akan ditugaskan untuk menjalankan pengujian Anda. Jika saat ini tidak ada instance yang tersedia, Device Farm akan menunggu instance ARN perangkat pertama yang tersedia, dan menetapkannya untuk menjalankan pengujian Anda.

Label instance perangkat

Label instance perangkat adalah pengenal tekstual yang dapat Anda lampirkan sebagai metadata untuk instance perangkat. Anda dapat melampirkan beberapa label ke setiap instance perangkat

dan label yang sama ke beberapa instance perangkat. Untuk informasi selengkapnya tentang menambahkan, memodifikasi, atau menghapus label perangkat dari instance perangkat, lihatMengelola perangkat pribadi.

Label instans perangkat dapat menjadi cara yang kuat untuk memilih perangkat pribadi untuk kumpulan perangkat karena, jika Anda memiliki beberapa instance perangkat dengan label yang sama, maka ini memungkinkan kumpulan perangkat untuk memilih salah satu dari mereka untuk pengujian Anda. Jika ARN perangkat bukan aturan yang baik untuk kasus penggunaan Anda (misalnya, jika Anda ingin memilih dari perangkat dari beberapa jenis perangkat, atau jika Anda ingin memilih dari subset semua perangkat jenis perangkat), maka label instance perangkat dapat memungkinkan Anda memilih dari beberapa perangkat untuk kumpulan perangkat Anda dengan perincian yang lebih besar. Instance perangkat pribadi individu dapat mengalami kegagalan perangkat keras, mendorong Device Farm untuk secara otomatis menggantinya dengan instance kerja baru dari jenis perangkat yang sama. Dalam skenario ini, instance perangkat pengganti tidak akan menyimpan metadata label instance apa pun dari perangkat yang diganti. Jadi, jika Anda menerapkan label instance perangkat yang sama ke beberapa instance perangkat, maka aturan label instance perangkat memastikan bahwa kumpulan perangkat Anda dapat terus memilih instance perangkat jika terjadi kegagalan perangkat keras.

Saat Anda menggunakan aturan label instance perangkat untuk perangkat pribadi di kumpulan perangkat Anda dan menjadwalkan pengujian yang dijalankan dengan kumpulan tersebut, Device Farm akan secara otomatis memeriksa instance perangkat pribadi mana yang diwakili oleh label instance perangkat tersebut, dan instance tersebut, pilih secara acak yang tersedia untuk menjalankan pengujian Anda. Jika tidak ada yang tersedia, Device Farm akan secara acak memilih instance perangkat apa pun dengan label instance perangkat untuk menjalankan pengujian Anda dan mengantre pengujian untuk dijalankan di perangkat setelah tersedia.

Contoh ARN

Sebuah instance perangkat ARN adalah pengidentifikasi yang mewakili instance perangkat logam kosong fisik yang digunakan dalam armada pribadi. Misalnya, jika Anda memiliki tiga*iPhone 13*perangkat di OS*15.0.0*di armada pribadi Anda, sementara setiap perangkat akan berbagi perangkat ARN yang sama, setiap perangkat juga akan memiliki ARN instans sendiri yang mewakili instance itu saja.

ARN instance perangkat adalah cara yang paling tidak kuat untuk memilih perangkat pribadi untuk kumpulan perangkat dan hanya disarankan jika ARN perangkat dan label instance perangkat tidak sesuai dengan kasus penggunaan Anda. ARN instance perangkat sering digunakan sebagai

aturan untuk kumpulan perangkat ketika instance perangkat tertentu dikonfigurasi dengan cara yang unik dan spesifik sebagai prasyarat untuk pengujian Anda dan jika konfigurasi tersebut perlu diketahui dan diverifikasi sebelum pengujian dijalankan di sana. Instance perangkat pribadi individu dapat mengalami kegagalan perangkat keras, mendorong Device Farm untuk secara otomatis menggantinya dengan instance kerja baru dari jenis perangkat yang sama. Dalam skenario ini, instance perangkat pengganti akan memiliki ARN instance perangkat yang berbeda dari perangkat yang diganti. Jadi, jika Anda mengandalkan ARN instance perangkat untuk kumpulan perangkat Anda, maka Anda harus mengubah definisi aturan kumpulan perangkat secara manual dari menggunakan ARN lama menjadi menggunakan ARN baru. Jika Anda perlu mengkonfigurasi perangkat secara manual untuk pengujiannya, maka ini bisa menjadi alur kerja yang efektif (dibandingkan dengan ARN perangkat). Untuk pengujian pada skala besar, disarankan untuk mencoba mengadaptasi kasus penggunaan ini agar berfungsi dengan label instans perangkat dan jika memungkinkan, memiliki beberapa instance perangkat yang telah dikonfigurasi sebelumnya untuk pengujian.

Saat Anda menggunakan aturan ARN instance perangkat untuk perangkat pribadi di kumpulan perangkat Anda dan menjadwalkan pengujian yang dijalankan dengan kumpulan tersebut, Device Farm akan secara otomatis menetapkan pengujian tersebut ke instance perangkat tersebut. Jika instance perangkat tersebut tidak tersedia, Device Farm akan mengantri pengujian pada perangkat setelah tersedia.

Membuat kolam perangkat pribadi dengan perangkat pribadi (konsol)

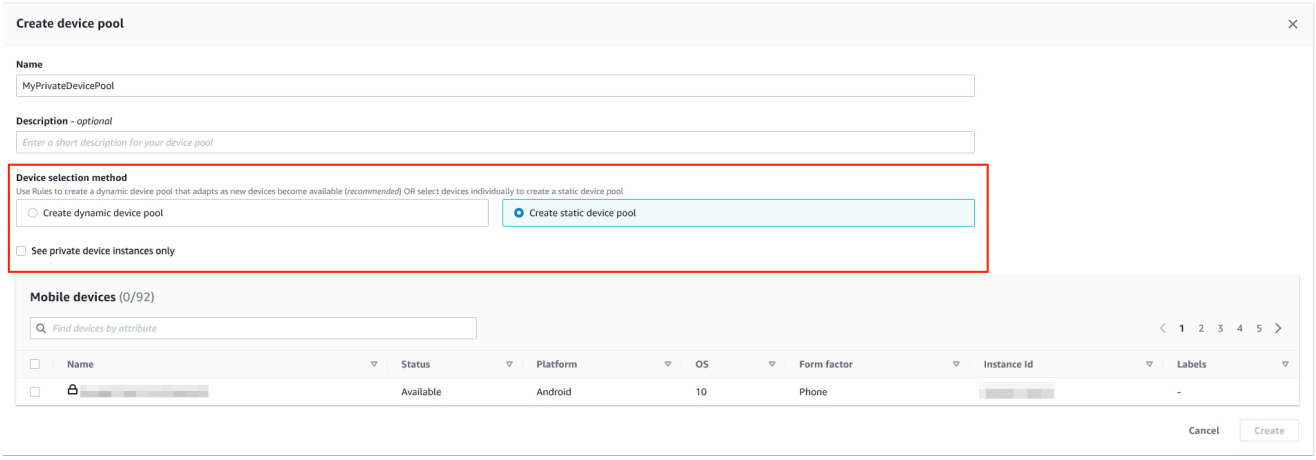
Saat membuat uji coba, Anda dapat membuat kumpulan perangkat untuk uji coba dan memastikan bahwa kumpulan tersebut hanya menyertakan perangkat pribadi Anda.

> **Note**
>
> Saat membuat kumpulan perangkat dengan perangkat pribadi di konsol, Anda hanya dapat menggunakan salah satu dari tiga aturan yang tersedia untuk memilih perangkat pribadi. Jika Anda ingin membuat kumpulan perangkat yang berisi beberapa jenis aturan untuk perangkat pribadi (misalnya, kumpulan perangkat yang berisi aturan untuk ARN perangkat dan ARN instance perangkat), maka Anda perlu membuat kumpulan melalui CLI atau API.

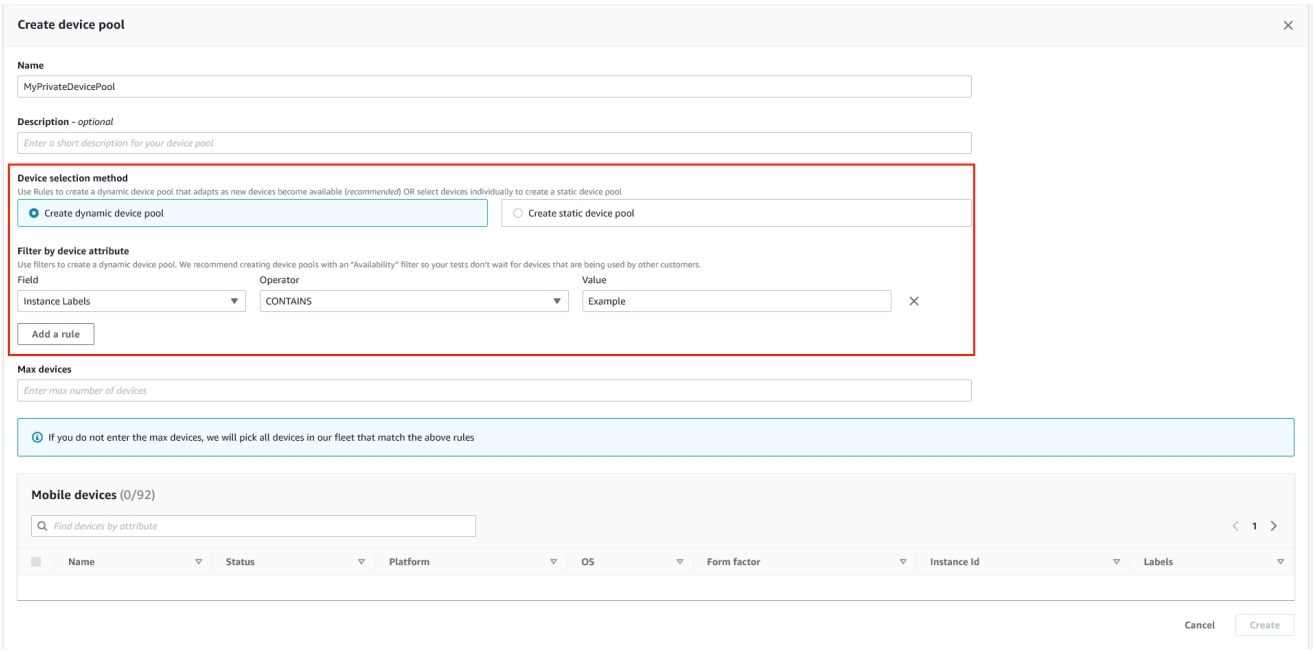
1. Buka konsol Device Farm dihttps://console.aws.amazon.com/devicefarm/.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.

3. Pilih proyek yang sudah ada dari daftar atau buat yang baru. Untuk membuat proyek baru, pilihProyek baru, masukkan nama untuk proyek, lalu pilihKirim.
4. PilihTes otomatis, dan kemudian pilihBuat run baru. Wizard memandu Anda melalui langkah-langkah untuk memilih aplikasi Anda dan mengonfigurasi pengujian yang ingin Anda jalankan.
5. UntukPilih perangkatlangkah, pilihBuat kumpulan perangkat, dan masukkan nama dan deskripsi opsional untuk kumpulan perangkat Anda.

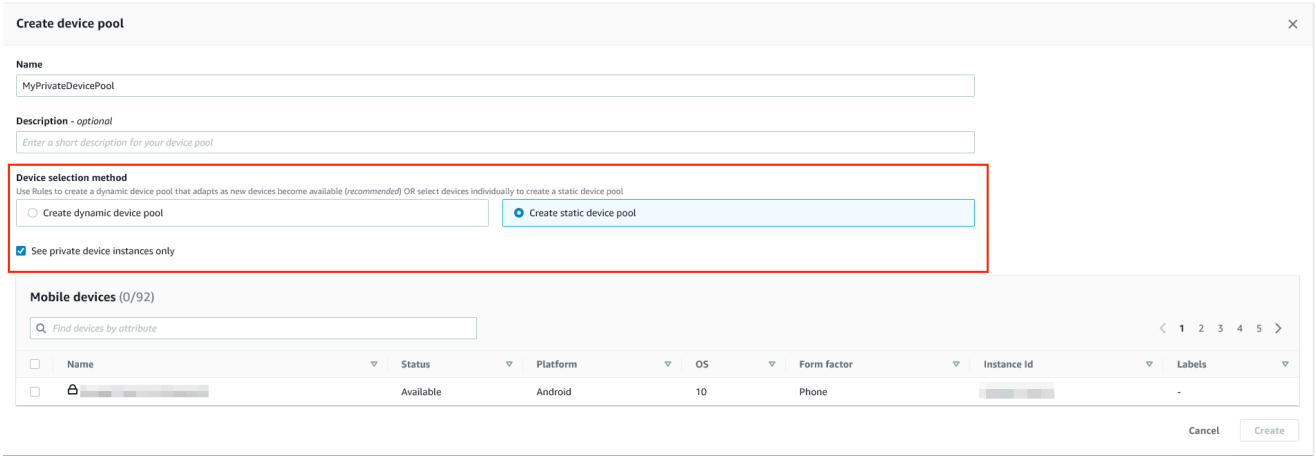
   a. Untuk menggunakan aturan ARN perangkat untuk kumpulan perangkat Anda, pilihBuat kolam perangkat statis, lalu pilih jenis perangkat tertentu dari daftar yang ingin Anda gunakan di kumpulan perangkat. Jangan pilihInstans perangkat pribadi sajakarena opsi ini menyebabkan kumpulan perangkat dibuat dengan aturan ARN instance perangkat (bukan aturan ARN perangkat).



   b. Untuk menggunakan aturan label instance perangkat untuk kumpulan perangkat Anda, pilihBuat kolam perangkat dinamis. Kemudian, untuk setiap label yang ingin Anda gunakan di kumpulan perangkat, pilihTambahkan aturan. Untuk setiap aturan, pilihLabel InstancesebagaiField, pilihBerisisebagaiOperator, dan tentukan label instance perangkat yang Anda inginkan sebagaiValue.



c. Untuk menggunakan aturan ARN instance perangkat untuk kumpulan perangkat Anda, pilihBuat kolam perangkat statis, lalu pilihInstans perangkat pribadi sajauntuk membatasi daftar perangkat hanya untuk instans perangkat pribadi yang telah dikaitkan dengan Device FarmAWSakun.



6. Pilih Create (Buat).

Membuat kolam perangkat pribadi dengan perangkat pribadi (AWS CLI)

- Jalankan perintah create-device-pool.

Untuk informasi tentang menggunakan Device Farm denganAWS CLI, lihatReferensi AWS CLI.

Membuat kumpulan perangkat pribadi dengan perangkat pribadi (API)

- PanggilCreateDevicePoolAPI.

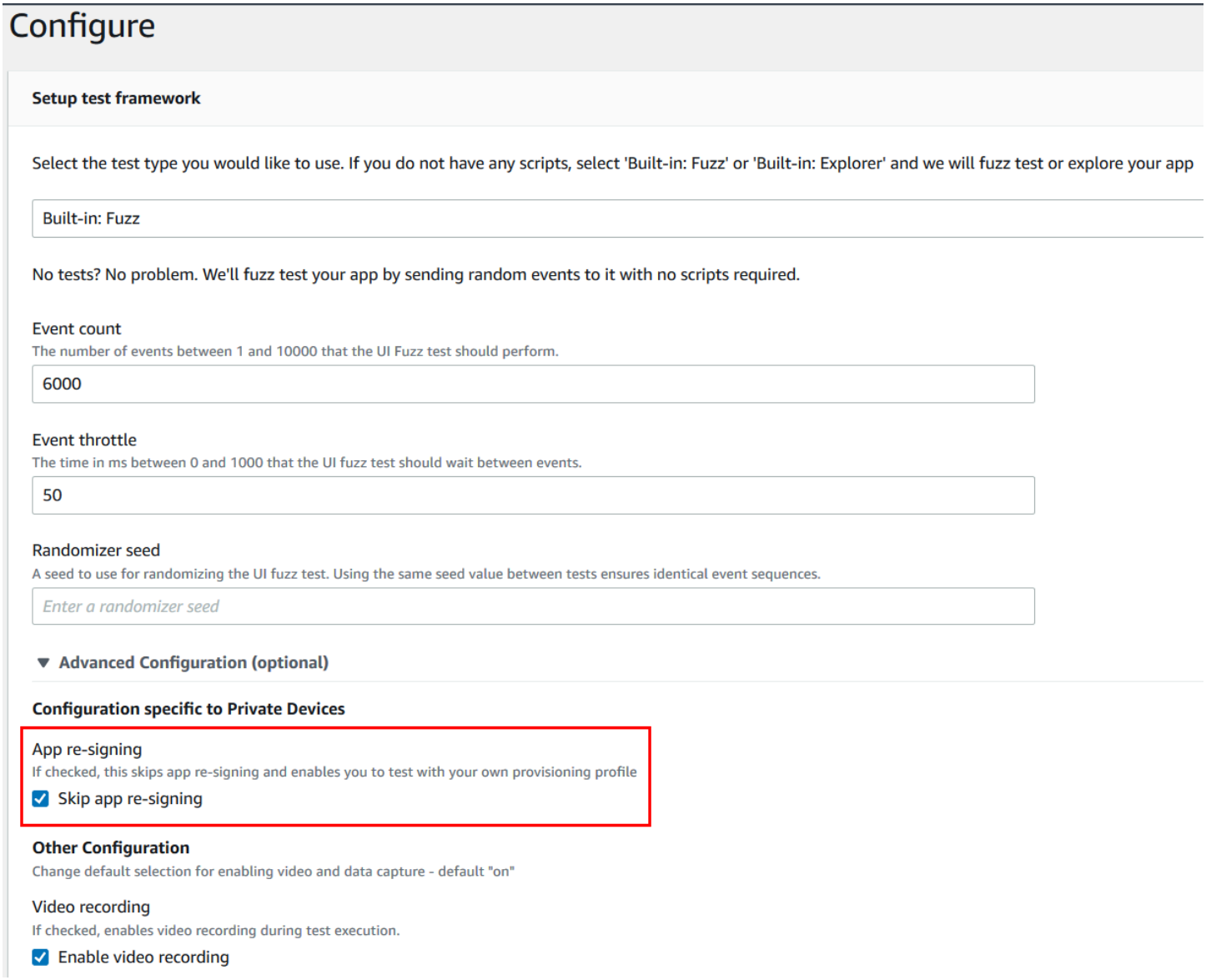
Untuk informasi tentang menggunakan Device Farm API, lihatMengotomatisasi Perangkat Pertanian.

Melewatkan penandatanganan ulang aplikasi pada perangkat pribadi di AWS Device Farm

Saat menggunakan perangkat pribadi, Anda dapat melewati langkah di mana AWS Device Farm menandatangani ulang aplikasi Anda. Ini berbeda dengan perangkat publik, di mana Device Farm selalu menandatangani ulang aplikasi Anda di platform Android dan iOS.

Anda dapat melewati penandatanganan ulang aplikasi saat membuat sesi akses jarak jauh atau uji coba. Ini dapat membantu jika aplikasi Anda memiliki fungsionalitas yang rusak saat Device Farm menandatangani ulang aplikasi Anda. Misalnya, pemberitahuan push mungkin tidak berfungsi setelah penandatanganan ulang. Untuk informasi selengkapnya tentang perubahan yang dilakukan Device Farm saat menguji aplikasi Anda, lihatFAQ AWS Device Farm.

Untuk melewati penandatanganan ulang aplikasi untuk uji coba, pilihLewati penandatanganan ulang aplikasipadaKonfigurasikanhalaman saat Anda membuat uji coba.



> **Note**
>
> Jika Anda menggunakan kerangka kerja XCTest,Lewati penandatanganan ulang aplikasipilihan tidak tersedia. Untuk informasi selengkapnya, lihat Bekerja dengan XCTest untuk iOS dan AWS Device Farm.

Langkah-langkah tambahan untuk mengonfigurasi setelan penandatanganan aplikasi bervariasi, tergantung apakah Anda menggunakan perangkat Android atau iOS pribadi.

Melewatkan penandatanganan ulang aplikasi di perangkat Android

Jika Anda menguji aplikasi di perangkat Android pribadi, pilihLewati penandatanganan ulang aplikasisaat Anda membuat uji coba atau sesi akses jarak jauh Anda. Tidak ada konfigurasi lain yang diperlukan.

Melewatkan penandatanganan ulang aplikasi di perangkat iOS

Apple mengharuskan Anda menandatangani aplikasi untuk pengujian sebelum Anda memuatnya ke perangkat. Untuk perangkat iOS, Anda memiliki dua opsi untuk menandatangani aplikasi.

- Jika Anda menggunakan profil pengembang internal (Enterprise), Anda dapat melompat ke bagian berikutnya,the section called “Membuat sesi akses jarak jauh untuk mempercayai aplikasi Anda”.

- Jika Anda menggunakan profil pengembangan aplikasi iOS ad hoc, Anda harus terlebih dahulu mendaftarkan perangkat dengan akun pengembang Apple Anda, lalu memperbarui profil penyediaan Anda untuk menyertakan perangkat pribadi. Anda kemudian harus menandatangani ulang aplikasi Anda dengan profil penyediaan yang Anda perbarui. Anda kemudian dapat menjalankan aplikasi yang ditandatangani ulang di Device Farm.

Untuk mendaftarkan perangkat dengan profil penyediaan pengembangan aplikasi iOS ad hoc

1. Masuk ke akun pengembang Apple Anda.
2. Arahkan keSertifikat, ID, dan Profilbagian dari konsol.
3. Pergi kePerangkat.
4. Daftarkan perangkat di akun pengembang Apple Anda. Untuk mendapatkan nama dan UDID perangkat, gunakan`ListDeviceInstances`pengoperasian API Device Farm.
5. Buka profil penyediaan Anda dan pilihSunting.
6. Pilih perangkat dari daftar.
7. Di Xcode, ambil profil penyediaan Anda yang diperbarui, lalu tandatangani ulang aplikasi.

Tidak ada konfigurasi lain yang diperlukan. Anda sekarang dapat membuat sesi akses jarak jauh atau uji coba dan pilihLewati penandatanganan ulang aplikasi.

Membuat sesi akses jarak jauh untuk mempercayai aplikasi iOS Anda

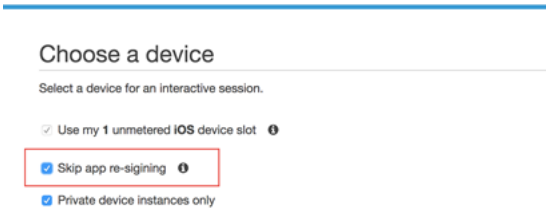
Jika Anda menggunakan profil penyediaan pengembang internal (Enterprise), Anda harus melakukan prosedur satu kali untuk mempercayai sertifikat pengembang aplikasi internal di setiap perangkat pribadi Anda.

Untuk melakukannya, Anda dapat menginstal aplikasi yang ingin Anda uji di perangkat pribadi, atau Anda dapat menginstal aplikasi tiruan yang ditandatangani dengan sertifikat yang sama dengan aplikasi yang ingin Anda uji. Ada keuntungan untuk menginstal aplikasi dummy yang ditandatangani dengan sertifikat yang sama. Setelah Anda mempercayai profil konfigurasi atau pengembang aplikasi perusahaan, semua aplikasi dari pengembang tersebut dipercaya di perangkat pribadi hingga Anda menghapusnya. Oleh karena itu, saat Anda mengunggah versi baru aplikasi yang ingin Anda uji, Anda tidak perlu mempercayai pengembang aplikasi lagi. Ini sangat berguna jika Anda menjalankan otomatisasi pengujian dan Anda tidak ingin membuat sesi akses jarak jauh setiap kali Anda menguji aplikasi Anda.

Sebelum Anda memulai sesi akses jarak jauh, ikuti langkah-langkah diMembuat profil instansuntuk membuat atau memodifikasi profil instance di Device Farm. Di profil instance, tambahkan ID bundel aplikasi pengujian atau aplikasi dummy keKecualikan paket dari pembersihanpengaturan. Kemudian, lampirkan profil instance ke instance perangkat pribadi untuk memastikan bahwa Device Farm tidak menghapus aplikasi ini dari perangkat sebelum memulai uji coba baru. Ini memastikan bahwa sertifikat pengembang Anda tetap tepercaya.

Anda dapat mengunggah aplikasi dummy ke perangkat dengan menggunakan sesi akses jarak jauh, yang memungkinkan Anda meluncurkan aplikasi dan mempercayai pengembang.

1. Ikuti instruksi diBuat sesiuntuk membuat sesi akses jarak jauh yang menggunakan profil instans perangkat pribadi yang Anda buat. Saat Anda membuat sesi, pastikan untuk memilihLewati penandatanganan ulang aplikasi.



> **Important**
>
> Untuk memfilter daftar perangkat yang hanya menyertakan perangkat pribadi, pilihInstans perangkat pribadi sajauntuk memastikan bahwa Anda menggunakan perangkat pribadi dengan profil instans yang benar.

Pastikan untuk juga menambahkan aplikasi dummy atau aplikasi yang ingin Anda uji keKecualikan paket dari pembersihanpengaturan untuk profil instance yang dilampirkan ke instance ini.

2. Saat sesi jarak jauh Anda dimulai, pilihPilih Fileuntuk menginstal aplikasi yang menggunakan profil penyediaan internal Anda.
3. Luncurkan aplikasi yang baru saja Anda unggah.
4. Ikuti instruksi untuk mempercayai sertifikat pengembang.

Semua aplikasi dari profil konfigurasi atau pengembang aplikasi perusahaan ini sekarang dipercaya di perangkat pribadi ini hingga Anda menghapusnya.

Menggunakan layanan endpoint Amazon VPC dengan Device Farm

> **Note**
>
> Menggunakan Amazon VPC Endpoint Services dengan Device Farm hanya didukung untuk pelanggan dengan perangkat pribadi yang dikonfigurasi. Untuk mengaktifkan akun AWS Anda menggunakan fitur ini dengan perangkat pribadi, silakanhubungi kami.

Amazon Virtual Private Cloud (Amazon VPC) adalahAWSLayanan yang dapat Anda gunakan untuk meluncurkanAWSsumber daya dalam jaringan virtual yang Anda tentukan. Dengan VPC, Anda memiliki kontrol atas pengaturan jaringan Anda, seperti rentang alamat IP, subnet, tabel routing, dan gateway jaringan.

Jika Anda menggunakan Amazon VPC untuk meng-host aplikasi pribadi di AS Barat (Oregon) (`us-west-2`)AWSWilayah, Anda dapat membuat koneksi pribadi antara VPC dan Device Farm Anda.

Dengan koneksi ini, Anda dapat menggunakan Device Farm untuk menguji aplikasi pribadi tanpa mengeksposnya melalui internet publik. Untuk mengaktifkanAWSakun untuk menggunakan fitur ini dengan perangkat pribadi,hubungi kami.

Untuk menghubungkan sumber daya di VPC ke Device Farm, Anda dapat menggunakan konsol Amazon VPC untuk membuat layanan titik akhir VPC. Layanan endpoint ini memungkinkan Anda menyediakan sumber daya di VPC ke Device Farm melalui titik akhir VPC Device Farm. Layanan endpoint menyediakan konektivitas yang andal dan dapat diskalakan ke Device Farm tanpa memerlukan gateway internet, instance terjemahan alamat jaringan (NAT), atau koneksi VPN. Untuk informasi lebih lanjut, lihatLayanan titik akhir VPC (AWSPrivateLink)diAWS PrivateLinkPanduan.

> **Important**
>
> Fitur titik akhir Device Farm VPC membantu Anda menghubungkan layanan internal pribadi dengan aman di VPC Anda ke VPC publik Device Farm dengan menggunakanAWS PrivateLinkkoneksi. Meskipun koneksi aman dan pribadi, keamanan itu tergantung pada perlindungan AndaAWSkredensialnya. Jika AndaAWSkredensialnya dikompromikan, penyerang dapat mengakses atau mengekspos data layanan Anda ke dunia luar.

Setelah membuat layanan titik akhir VPC di Amazon VPC, Anda dapat menggunakan konsol Device Farm untuk membuat konfigurasi titik akhir VPC di Device Farm. Topik ini menunjukkan cara membuat koneksi Amazon VPC dan konfigurasi titik akhir VPC di Device Farm.

Sebelum Anda memulai

Informasi berikut adalah untuk pengguna Amazon VPC di AS Barat (Oregon) (`us-west-2`) Wilayah, dengan subnet di masing-masing Availability Zone berikut: us-west-2a, us-west-2b, dan us-west-2c.

Device Farm memiliki persyaratan tambahan untuk layanan endpoint VPC yang dapat Anda gunakan. Saat membuat dan mengonfigurasi layanan titik akhir VPC agar berfungsi dengan Device Farm, pastikan Anda memilih opsi yang memenuhi persyaratan berikut:

- Availability Zone untuk layanan harus mencakup us-west-2a, us-west-2b, dan us-west-2c. Network Load Balancer yang terkait dengan layanan endpoint VPC menentukan Availability Zones untuk layanan endpoint VPC tersebut. Jika layanan titik akhir VPC Anda tidak menampilkan ketiga Availability Zone ini, Anda harus membuat ulang Network Load Balancer untuk mengaktifkan ketiga zona ini, lalu mengasosiasikan kembali Network Load Balancer dengan layanan endpoint Anda.

- Prinsipal yang diizinkan untuk layanan endpoint harus menyertakan Amazon Resource Name (ARN) dari titik akhir Device Farm VPC (service ARN). Setelah Anda membuat layanan endpoint Anda, tambahkan layanan titik akhir Device Farm VPC ARN ke daftar izin Anda untuk memberikan izin Device Farm untuk mengakses layanan titik akhir VPC Anda. Untuk mendapatkan layanan titik akhir Device Farm VPC ARN,hubungi kami.

Selain itu, jika Anda menyimpanPenerimaan diperlukanpengaturan diaktifkan saat Anda membuat layanan titik akhir VPC, Anda harus secara manual menerima setiap permintaan koneksi yang dikirim Device Farm ke layanan endpoint. Untuk mengubah setelan ini untuk layanan endpoint yang ada, pilih layanan endpoint di konsol Amazon VPC, pilihTindakan, dan kemudian pilihUbah pengaturan penerimaan titik akhir. Untuk informasi lebih lanjut, lihatUbah penyeimbang beban dan pengaturan penerimaandiAWS PrivateLinkPanduan.

Bagian selanjutnya menjelaskan cara membuat layanan endpoint Amazon VPC yang memenuhi persyaratan ini.

Langkah 1: Membuat Network Load Balancer

Langkah pertama dalam membangun koneksi pribadi antara VPC dan Device Farm Anda adalah membuat Network Load Balancer untuk merutekan permintaan ke grup target.

New console

Untuk membuat Network Load Balancer menggunakan konsol baru

1. Buka konsol Amazon Elastic Compute Cloud (Amazon EC2) dihttps://console.aws.amazon.com/ec2/.
2. Di panel navigasi, di bawahPenyeimbangan beban, pilihPenyeimbang beban.
3. Pilih Buat Penyeimbang Beban.
4. Di bawahPenyeimbang beban jaringan, pilihBuat.
5. PadaBuat penyeimbang beban jaringanhalaman, di bawahKonfigurasi dasar, lakukan hal berikut:
   a. Masukkan penyeimbang bebanNama.
   b. UntukSkema, pilihInternal.
6. Di bawahPemetaan jaringan, lakukan hal berikut:
   a. PilihVPCUntuk kelompok target Anda.

b. Pilih yang berikutPemetaan:

- `us-west-2a`
- `us-west-2b`
- `us-west-2c`

7. Di bawahPendengar dan perutean, gunakanProtokoldanPelabuhanpilihan untuk memilih kelompok target Anda.

> **Note**
>
> Secara default, penyeimbangan beban zona ketersediaan silang dinonaktifkan. Karena penyeimbang beban menggunakan Availability Zones`us-west-2a`,`us-west-2b`, dan`us-west-2c`, itu mengharuskan target untuk didaftarkan di masing-masing Availability Zone tersebut, atau, jika Anda mendaftarkan target di kurang dari ketiga zona, Anda mengharuskan Anda mengaktifkan penyeimbangan beban lintas zona. Jika tidak, penyeimbang beban mungkin tidak berfungsi seperti yang diharapkan.

8. Pilih Buat Penyeimbang Beban.

Old console

Untuk membuat Network Load Balancer menggunakan konsol lama

1. Buka konsol Amazon Elastic Compute Cloud (Amazon EC2) dihttps://console.aws.amazon.com/ec2/.
2. Di panel navigasi, di bawahPenyeimbangan beban, pilihpenyeimbang beban.
3. Pilih Buat Penyeimbang Beban.
4. Di bawahPenyeimbang beban jaringan, pilihBuat.
5. PadaKonfigurasikan penyeimbang bebanhalaman, di bawahKonfigurasi dasar, lakukan hal berikut:

   a. Masukkan penyeimbang bebanNama.

   b. UntukSkema, pilihInternal.

6. Di bawahPendengar, pilihProtokoldanPelabuhanyang digunakan kelompok target Anda.
7. Di bawahZona ketersediaan, lakukan hal berikut:

a. PilihVPCUntuk kelompok target Anda.

b. Pilih yang berikutZona ketersediaan:

- `us-west-2a`
- `us-west-2b`
- `us-west-2c`

c. PilihBerikutnya: mengkonfigurasi pengaturan keamanan.

8. (Opsional) Konfigurasikan pengaturan keamanan Anda, lalu pilihBerikutnya: mengkonfigurasi routing.
9. Pada halaman Configure Routing, lakukan hal berikut:

a. UntukKelompok sasaran, pilihKelompok sasaran yang ada.

b. UntukNama, pilih grup target Anda.

c. PilihBerikutnya: daftar target.

10. PadaDaftarkan targethalaman, tinjau target Anda, lalu pilihBerikutnya: ulasan.

> **Note**
>
> Secara default, penyeimbangan beban zona ketersediaan silang dinonaktifkan. Karena penyeimbang beban menggunakan Availability Zones`us-west-2a`,`us-west-2b`, dan`us-west-2c`, itu mengharuskan target untuk didaftarkan di masing-masing Availability Zone tersebut, atau, jika Anda mendaftarkan target di kurang dari ketiga zona, Anda mengharuskan Anda mengaktifkan penyeimbangan beban lintas zona. Jika tidak, penyeimbang beban mungkin tidak berfungsi seperti yang diharapkan.

11. Tinjau konfigurasi penyeimbang beban Anda, lalu pilihBuat.

Langkah 2: Membuat layanan endpoint Amazon VPC

Setelah membuat Network Load Balancer, gunakan konsol Amazon VPC untuk membuat layanan endpoint di VPC Anda.

1. Buka konsol Amazon VPC di https://console.aws.amazon.com/vpc/.
2. Di bawahSumber daya menurut wilayah, pilihLayanan titik akhir.

3. PilihBuat layanan endpoint.
4. Lakukan salah satu dari berikut:

   - Jika Anda sudah memiliki Network Load Balancer yang ingin digunakan layanan endpoint, pilih di bawahPenyeimbang beban yang tersedia, dan kemudian lanjutkan ke langkah 5.
   - Jika Anda belum membuat Network Load Balancer, pilihBuat penyeimbang beban baru. Konsol Amazon EC2 terbuka. Ikuti langkah-langkahnya diMembuat Network Load Balancerdimulai dengan langkah 3, lalu lanjutkan dengan langkah-langkah ini di konsol Amazon VPC.

5. UntukTermasuk zona ketersediaan, verifikasi bahwa`us-west-2a`,`us-west-2b`, dan`us-west-2c`muncul dalam daftar.
6. Jika Anda tidak ingin secara manual menerima atau menolak setiap permintaan koneksi yang dikirim ke layanan endpoint, di bawahPengaturan tambahan, jelasPenerimaan diperlukan. Jika Anda menghapus kotak centang ini, layanan endpoint secara otomatis menerima setiap permintaan koneksi yang diterimanya.
7. Pilih Create (Buat).
8. Di layanan endpoint baru, pilihIzinkan kepala sekolah.
9. Hubungi kamiuntuk mendapatkan ARN dari titik akhir VPC Device Farm (layanan ARN) untuk ditambahkan ke daftar izinkan untuk layanan titik akhir, dan kemudian tambahkan ARN layanan itu ke daftar izinkan untuk layanan tersebut.
10. PadaDetailtab untuk layanan titik akhir, buat catatan nama layanan (nama layanan). Anda memerlukan nama ini saat membuat konfigurasi titik akhir VPC di langkah berikutnya.

Layanan titik akhir VPC Anda sekarang siap digunakan dengan Device Farm.

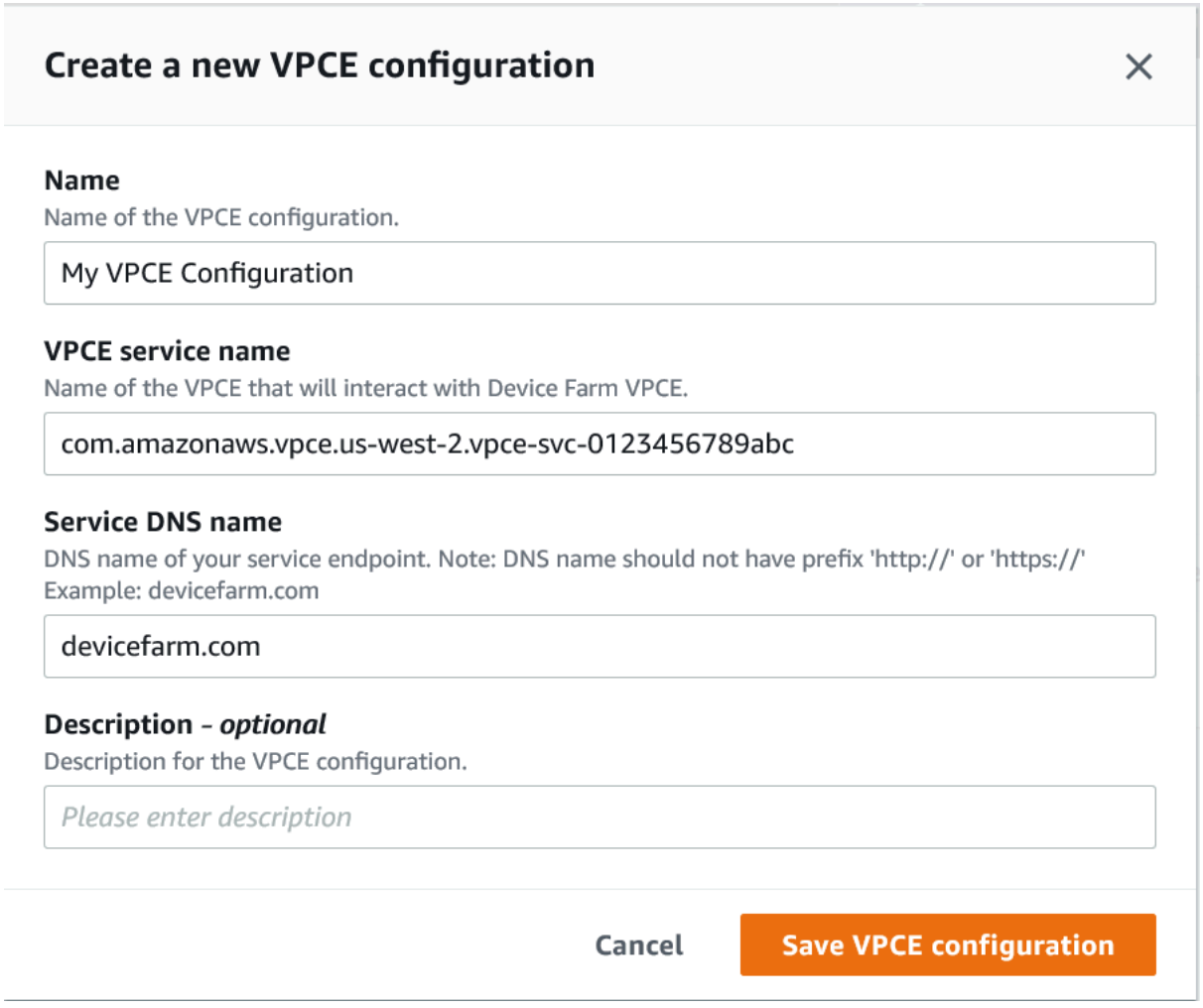
Langkah 3: Membuat konfigurasi titik akhir VPC di Device Farm

Setelah membuat layanan endpoint di Amazon VPC, Anda dapat membuat konfigurasi endpoint Amazon VPC di Device Farm.

1. Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.
2. Di panel navigasi, pilihPengujian perangkat seluler, makaPerangkat pribadi.
3. PilihKonfigurasi VPCE.
4. PilihBuat konfigurasi VPCE.

5. Di bawahBuat konfigurasi VPCE baru, masukkanNamauntuk konfigurasi titik akhir VPC.
6. UntukNama layanan VPCE, masukkan nama layanan titik akhir Amazon VPC (nama layanan) yang Anda catat di konsol Amazon VPC. Namanya terlihat seperti`com.amazonaws.vpce.us-west-2.vpce-svc-id`.
7. UntukNama DNS layanan, masukkan nama DNS layanan untuk aplikasi yang ingin Anda uji (misalnya,`devicefarm.com`). Jangan tentukan`http`atau`https`sebelum nama DNS layanan.

   Nama domain tidak dapat diakses melalui internet publik. Selain itu, nama domain baru ini, yang dipetakan ke layanan endpoint VPC Anda, dihasilkan oleh Amazon Route 53 dan tersedia secara eksklusif untuk Anda di sesi Device Farm Anda.
8. Pilih Save (Simpan).



Langkah 4: Membuat uji coba

Setelah menyimpan konfigurasi titik akhir VPC, Anda dapat menggunakan konfigurasi untuk membuat pengujian berjalan atau mengakses sesi dari jarak jauh. Untuk informasi lebih lanjut, lihat Membuat uji coba di Device Farm atau Buat sesi.

Bekerja dengan Amazon VPC di seluruhAWSDaerah

Layanan Device Farm hanya berlokasi di AS Barat (Oregon) (`us-west-2`) Wilayah. Anda dapat menggunakan Amazon Virtual Private Cloud (Amazon VPC) untuk menjangkau layanan di Amazon Virtual Private Cloud Anda di tempat lainAWSWilayah menggunakan Device Farm. Jika Device Farm dan layanan Anda berada di Wilayah yang sama, lihatMenggunakan layanan endpoint Amazon VPC dengan Device Farm.

Ada dua cara untuk mengakses layanan pribadi Anda yang berlokasi di Wilayah yang berbeda. Jika Anda memiliki layanan yang berlokasi di Wilayah lain yang tidak`us-west-2`, Anda dapat menggunakan VPC Peering untuk mengintip VPC Wilayah itu ke VPC lain yang berinteraksi dengan Device Farm di`us-west-2`. Namun, jika Anda memiliki layanan di beberapa Wilayah, Transit Gateway akan memungkinkan Anda mengakses layanan tersebut dengan konfigurasi jaringan yang lebih sederhana.

Untuk informasi lebih lanjut, lihatSkenario peering VPCdiPanduan Peering Amazon VPC.

Peering VPC

Anda dapat mengintip dua VPC mana pun di Wilayah yang berbeda, selama mereka memiliki blok CIDR yang berbeda dan tidak tumpang tindih. Ini memastikan bahwa semua alamat IP pribadi unik, dan memungkinkan semua sumber daya di VPC untuk saling menangani tanpa perlu bentuk terjemahan alamat jaringan (NAT) apa pun. Untuk informasi selengkapnya tentang notasi CIDR, lihat RFC 4632.

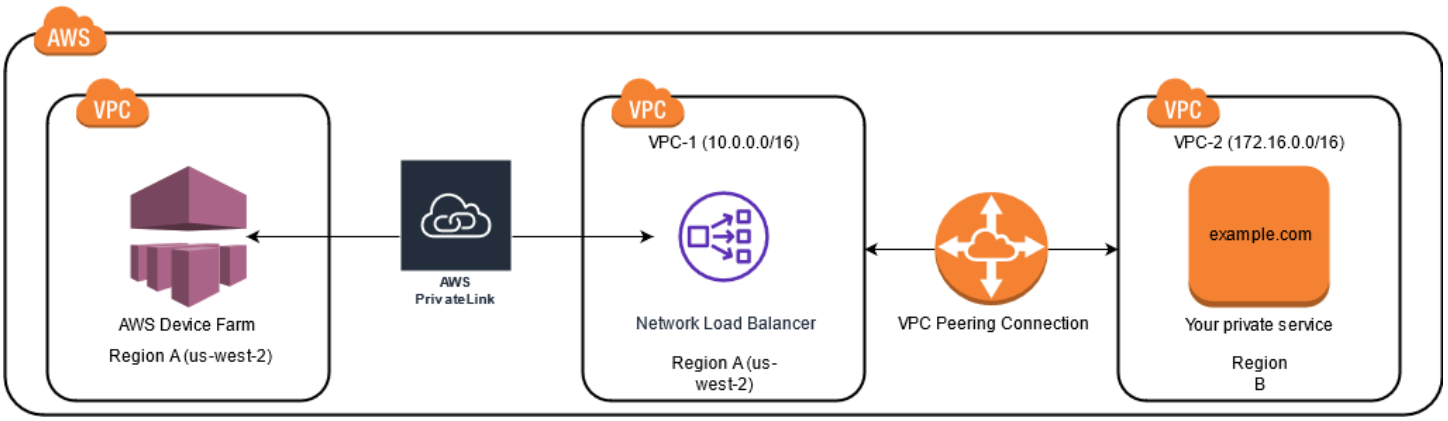
Topik ini mencakup skenario contoh lintas wilayah di mana Device Farm (disebut sebagaiVPC-1) terletak di AS Barat (Oregon) (`us-west-2`) Wilayah. VPC kedua dalam contoh ini (disebut sebagaiVPC-2) berada di wilayah lain.

Contoh Device Farm VPC Lintas wilayah

Komponen VPC	VPC-1	VPC-2
CIDR	10.0.0.0/16	172.16.0.0/16

> **Important**
>
> Membangun koneksi peering antara dua VPC dapat mengubah postur keamanan VPC. Selain itu, menambahkan entri baru ke tabel rute mereka dapat mengubah postur keamanan sumber daya dalam VPC. Adalah tanggung jawab Anda untuk menerapkan konfigurasi ini dengan cara yang memenuhi persyaratan keamanan organisasi Anda. Untuk informasi lebih lanjut, silakan lihatModel tanggung jawab bersama.

Diagram berikut menunjukkan komponen dalam contoh dan interaksi antara komponen-komponen ini.



Topik

- Prasyarat
- Langkah 1: Siapkan koneksi peering antara VPC-1 dan VPC-2
- Langkah 2: Perbarui tabel rute di VPC-1 dan VPC-2
- Langkah 3: Buat grup target
- Langkah 4: Buat Network Load Balancer
- Langkah 5: Buat layanan titik akhir VPC
- Langkah 6: Buat konfigurasi titik akhir VPC di Device Farm
- Langkah 7: Buat uji coba
- Buat jaringan yang dapat diskalakan dengan Transit Gateway

Prasyarat

Contoh ini membutuhkan yang berikut:

- Dua VPC yang dikonfigurasi dengan subnet yang berisi blok CIDR yang tidak tumpang tindih.
- VPC-1harus di`us-west-2`Wilayah dan berisi subnet untuk Availability Zones`us-west-2a`,`us-west-2b`, dan`us-west-2c`.

Untuk informasi selengkapnya tentang membuat VPC dan mengonfigurasi subnet, lihatBekerja dengan VPC dan subnetdiPanduan Peering Amazon VPC.

Langkah 1: Siapkan koneksi peering antara VPC-1 dan VPC-2

Buat koneksi peering antara dua VPC yang berisi blok CIDR yang tidak tumpang tindih. Untuk melakukan ini, lihatBuat dan terima koneksi peering VPCdiPanduan Peering Amazon VPC. Menggunakan skenario lintas wilayah topik ini danPanduan Peering Amazon VPC, contoh konfigurasi koneksi peering berikut dibuat:

Nama

```
Device-Farm-Peering-Connection-1
```

ID VPC (Pemohon)

```
vpc-0987654321gfedcba (VPC-2)
```

Akun

```
My account
```

Wilayah

```
US West (Oregon) (us-west-2)
```

ID VPC (Penerima)

```
vpc-1234567890abcdefg (VPC-1)
```

> **Note**
>
> Pastikan Anda berkonsultasi dengan kuota koneksi peering VPC Anda saat membuat koneksi peering baru. Untuk informasi lebih lanjut, silakan lihatKuota Amazon VPCdiPanduan Peering Amazon VPC.

Langkah 2: Perbarui tabel rute di VPC-1 dan VPC-2

Setelah mengatur koneksi peering, Anda harus membuat rute tujuan antara dua VPC untuk mentransfer data di antara mereka. Untuk menetapkan rute ini, Anda dapat memperbarui tabel rute secara manualVPC-1untuk menunjuk ke subnet dariVPC-2dan sebaliknya. Untuk melakukan ini, lihatPerbarui tabel rute Anda untuk koneksi peering VPCdiPanduan Peering Amazon VPC. Menggunakan skenario lintas wilayah topik ini danPanduan Peering Amazon VPC, contoh konfigurasi tabel rute berikut dibuat:

Contoh tabel rute VPC Device Farm

Komponen VPC	VPC-1	VPC-2
ID tabel rute	rtb-1234567890abcdefg	rtb-0987654321gfedcba
Rentang alamat lokal	10.0.0.0/16	172.16.0.0/16
Rentang alamat tujuan	172.16.0.0/16	10.0.0.0/16

Langkah 3: Buat grup target

Setelah mengatur rute tujuan, Anda dapat mengonfigurasi Network Load Balancer diVPC-1untuk merutekan permintaan keVPC-2.

Network Load Balancer harus terlebih dahulu berisi grup target yang berisi alamat IP tempat permintaan dikirim.

Untuk membuat grup target

1. Identifikasi alamat IP layanan yang ingin Anda targetkanVPC-2.
   - Alamat IP ini harus menjadi anggota subnet yang digunakan dalam koneksi peering.
   - Alamat IP yang ditargetkan harus statis dan tidak dapat diubah. Jika layanan Anda memiliki alamat IP dinamis, pertimbangkan untuk menargetkan sumber daya statis (seperti Network Load Balancer) dan meminta rute sumber daya statis tersebut ke target Anda yang sebenarnya.

> **Note**
>
> - Jika Anda menargetkan satu atau beberapa instans Amazon Elastic Compute Cloud (Amazon EC2) yang berdiri sendiri, buka konsol Amazon EC2 dihttps://console.aws.amazon.com/ec2/, lalu pilihContoh.
> - Jika Anda menargetkan grup Amazon EC2 Auto Scaling dari instans Amazon EC2, Anda harus mengaitkan grup Amazon EC2 Auto Scaling ke Network Load Balancer. Untuk informasi selengkapnya, lihat Memasang load balancer ke grup Auto Scaling Anda dalam Amazon EC2 Auto Scaling User Guide.
>
> Kemudian, Anda dapat membuka konsol Amazon EC2 dihttps://console.aws.amazon.com/ec2/, dan kemudian pilihAntarmuka Jaringan. Dari sana Anda dapat melihat alamat IP untuk masing-masing antarmuka jaringan Network Load Balancer di masing-masingZona Ketersediaan.

2. Buat grup target diVPC-1. Untuk melakukan ini, lihatBuat grup target untuk Network Load Balancer AndadiPanduan Pengguna untuk Network Load Balancers.

   Grup sasaran untuk layanan di VPC yang berbeda memerlukan konfigurasi berikut:

   - UntukPilih jenis target, pilihAlamat IP.
   - UntukVPC, pilih VPC yang akan menjadi tuan rumah penyeimbang beban. Untuk contoh topik, ini akanVPC-1.
   - PadaDaftarkan targethalaman, daftarkan target untuk setiap alamat IP diVPC-2.

     UntukJaringan, pilihAlamat IP pribadi lainnya.

     UntukZona Ketersediaan, pilih zona yang Anda inginkan diVPC-1.

     UntukAlamat IPv4, pilihVPC-2Alamat IP.

     UntukPelabuhan, pilih port Anda.
   - PilihSertakan sebagai tertunda di bawah ini. Setelah selesai menentukan alamat, pilihDaftarkan target yang tertunda.

Menggunakan skenario lintas wilayah topik ini danPanduan Pengguna untuk Network Load Balancers, nilai-nilai berikut digunakan dalam konfigurasi kelompok target:

Jenis target

```
IP addresses
```

Nama grup sasaran

```
my-target-group
```

Protokol/Pelabuhan

```
TCP : 80
```

VPC

```
vpc-1234567890abcdefg (VPC-1)
```

Jaringan

```
Other private IP address
```

Zona Ketersediaan

```
all
```

Alamat IPv4

```
172.16.100.60
```

Pelabuhan

```
80
```

Langkah 4: Buat Network Load Balancer

Buat Network Load Balancer menggunakan grup target yang dijelaskan dalamlangkah 3. Untuk melakukan ini, lihatMembuat Network Load Balancer.

Menggunakan skenario lintas wilayah topik ini, nilai berikut digunakan dalam contoh konfigurasi Network Load Balancer:

Nama penyeimbang beban

```
my-nlb
```

Skema

```
Internal
```

VPC

```
vpc-1234567890abcdefg (VPC-1)
```

Pemetaan

```
us-west-2a - subnet-4i23iuufkdiufsloi

us-west-2b - subnet-7x989pkjj78nmn23j

us-west-2c - subnet-0231ndmas12bnnsds
```

Protokol/Pelabuhan

```
TCP : 80
```

Kelompok Sasaran

```
my-target-group
```

Langkah 5: Buat layanan titik akhir VPC

Anda dapat menggunakan Network Load Balancer untuk membuat layanan endpoint VPC. Melalui layanan titik akhir VPC ini, Device Farm dapat terhubung ke layanan AndaVPC-2tanpa infrastruktur tambahan, seperti gateway internet, instans NAT, atau koneksi VPN.

Untuk melakukan ini, lihatMembuat layanan endpoint Amazon VPC.

Langkah 6: Buat konfigurasi titik akhir VPC di Device Farm

Sekarang Anda dapat membuat koneksi pribadi antara VPC dan Device Farm Anda. Anda dapat menggunakan Device Farm untuk menguji layanan pribadi tanpa mengeksposnya melalui internet publik. Untuk melakukan ini, lihatMembuat konfigurasi titik akhir VPC di Device Farm.

Menggunakan skenario lintas wilayah topik ini, nilai berikut digunakan dalam contoh konfigurasi titik akhir VPC:

Nama

```
My VPCE Configuration
```

Nama layanan VPCE

```
com.amazonaws.vpce.us-west-2.vpce-svc-1234567890abcdefg
```

Nama DNS layanan

```
devicefarm.com
```

Langkah 7: Buat uji coba

Anda dapat membuat uji coba yang menggunakan konfigurasi titik akhir VPC yang dijelaskan dalamlangkah 6. Untuk informasi lebih lanjut, lihat Membuat uji coba di Device Farm atau Buat sesi.

Buat jaringan yang dapat diskalakan dengan Transit Gateway

Untuk membuat jaringan yang dapat diskalakan menggunakan lebih dari dua VPC, Anda dapat menggunakan Transit Gateway untuk bertindak sebagai hub transit jaringan untuk menghubungkan VPC dan jaringan lokal Anda. Untuk mengonfigurasi VPC di wilayah yang sama dengan Device Farm untuk menggunakan Transit Gateway, Anda dapat mengikutiLayanan titik akhir Amazon VPC dengan Device Farmpanduan untuk menargetkan sumber daya di wilayah lain berdasarkan alamat IP pribadi mereka.

Untuk informasi selengkapnya tentang Transit Gateway, lihatApa itu gateway transit?diPanduan Gerbang Transit Amazon VPC.

Mengakhiri perangkat pribadi

> **Important**
>
> Petunjuk ini hanya berlaku untuk mengakhiri perjanjian perangkat pribadi. Untuk semua masalah AWS layanan dan penagihan lainnya, lihat dokumentasi masing-masing untuk produk tersebut atau hubungi AWS dukungan.

```
<Untuk mengakhiri perangkat pribadi setelah jangka waktu awal yang disepakati, A
```

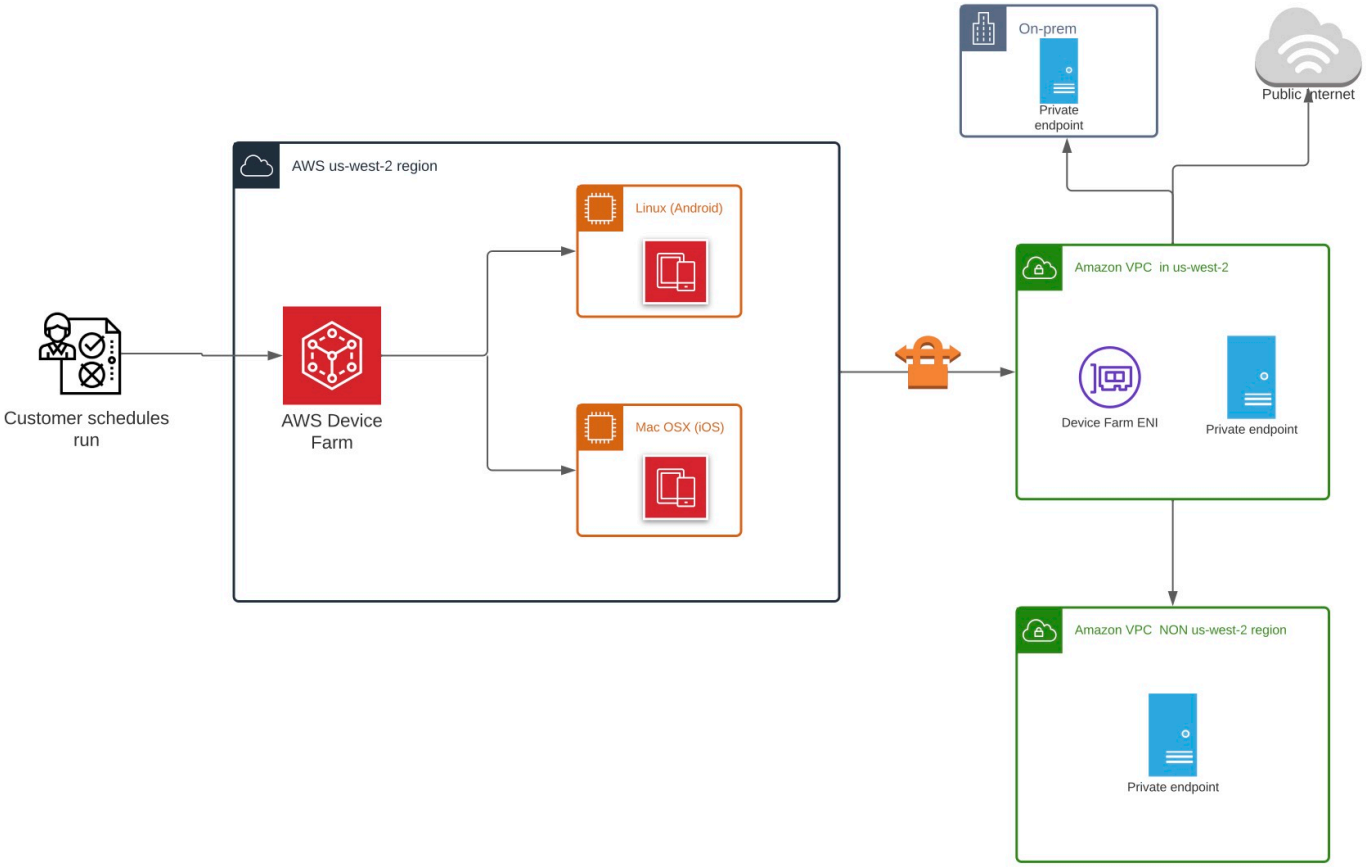
VPC-ENI di AWS Device Farm

> **Warning**
>
> Fitur ini hanya tersedia diperangkat pribadi. Untuk meminta penggunaan perangkat pribadi diAWSakun, silakanhubungi kami. Jika Anda sudah memiliki perangkat pribadi yang ditambahkan keAWSakun, kami sangat menyarankan menggunakan metode konektivitas VPC ini.

Fitur konektivitas VPC-ENI AWS Device Farm membantu pelanggan terhubung dengan aman ke titik akhir pribadi mereka yang dihostingAWS, perangkat lunak on-premise, atau penyedia cloud lainnya.

Anda dapat menghubungkan perangkat seluler Device Farm dan mesin hostnya ke lingkungan Amazon Virtual Private Cloud (Amazon VPC) di`us-west-2`Wilayah, yang memungkinkan akses ke terisolasi,non-internet-facinglayanan dan aplikasi melaluiantarmuka jaringan elastis. Untuk informasi lebih lanjut tentang VPC, lihatPanduan Pengguna Amazon VPC.

Jika titik akhir pribadi atau VPC Anda tidak ada di`us-west-2`Wilayah, Anda dapat menautkannya dengan VPC di`us-west-2`Wilayah menggunakan solusi sepertiGerbang TransitatauPengintip VPC. Dalam situasi seperti itu, Device Farm akan membuat ENI di subnet yang Anda sediakan untuk`us-west-2`Wilayah VPC, dan Anda akan bertanggung jawab untuk memastikan bahwa koneksi dapat dibuat antara`us-west-2`Wilayah VPC dan VPC di Wilayah lain.



Untuk informasi tentang penggunaanAWS CloudFormationuntuk secara otomatis membuat dan mengintip VPC, lihatTemplat VPCpeeringdiAWS CloudFormationrepositori template padaGitHub.

> **Note**
>
> Device Farm tidak mengenakan biaya apa pun untuk membuat ENI di VPC pelanggan di`us-west-2`. Biaya untuk konektivitas antar VPC lintas wilayah atau eksternal tidak termasuk dalam fitur ini.

Setelah Anda mengonfigurasi akses VPC, perangkat dan mesin host yang Anda gunakan untuk pengujian Anda tidak akan dapat terhubung ke sumber daya di luar VPC (misalnya, CDN publik)

kecuali ada gateway NAT yang Anda tentukan dalam VPC. Untuk informasi lebih lanjut, lihat Gateway NAT dalam Panduan Pengguna Amazon VPC.

Topik

AWSkontrol akses dan IAM

AWS Device Farm memungkinkan Anda untuk menggunakanAWS Identity and Access Management(IAM) untuk membuat kebijakan yang memberikan atau membatasi akses ke fitur Device Farm. Untuk menggunakan fitur Konektivitas VPC dengan AWS Device Farm, Kebijakan IAM berikut diperlukan untuk akun pengguna atau peran yang Anda gunakan untuk mengakses AWS Device Farm:

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": [
        "devicefarm:*",
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/devicefarm.amazonaws.com/
AWSServiceRoleForDeviceFarm",
      "Condition": {
```

```
        "StringLike": {
          "iam:AWSServiceName": "devicefarm.amazonaws.com"
        }
      }
    }
  ]
}
```

Untuk membuat atau memperbarui proyek Device Farm dengan konfigurasi VPC, kebijakan IAM Anda harus mengizinkan Anda memanggil tindakan berikut terhadap sumber daya yang tercantum dalam konfigurasi VPC:

```
"ec2:DescribeVpcs"
"ec2:DescribeSubnets"
"ec2:DescribeSecurityGroups"
"ec2:CreateNetworkInterface"
```

Selain itu, kebijakan IAM Anda juga harus mengizinkan pembuatan peran terkait layanan:

```
"iam:CreateServiceLinkedRole"
```

> **Note**
>
> Tak satu pun dari izin ini diperlukan untuk pengguna yang tidak menggunakan konfigurasi VPC dalam proyek mereka.

Peran terkait layanan

Penggunaan AWS Device FarmAWS Identity and Access Management(SAYA)peran terkait layanan. Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke Device Farm. Peran terkait layanan telah ditentukan sebelumnya oleh Device Farm dan menyertakan semua izin yang diperlukan layanan untuk memanggil lainnyaAWSlayanan atas nama Anda.

Peran terkait layanan membuat pengaturan Device Farm lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Device Farm mendefinisikan izin peran terkait layanan, dan kecuali ditentukan lain, hanya Device Farm yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Device Farm karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran yang terhubung dengan layanan, lihat Layanan AWS yang Berfungsi dengan IAM dan cari layanan yang memiliki Ya di kolom Peran yang Terhubung dengan Layanan. Pilih Ya dengan tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Izin peran terkait layanan untuk Device Farm

Device Farm menggunakan peran terkait layanan bernamaAWSServiceRoleForDeviceFarm—Memungkinkan Device Farm mengakses sumber daya AWS atas nama Anda.

AWSServiceRoleForDeviceFarm peran terkait layanan memercayakan layanan berikut untuk menjalankan peran tersebut:

- `devicefarm.amazonaws.com`

Kebijakan izin peran memungkinkan Device Farm menyelesaikan tindakan berikut:

- Untuk akun Anda
  - Buat antarmuka jaringan
  - Jelaskan antarmuka jaringan
  - Jelaskan VPC
  - Jelaskan subnet
  - Jelaskan kelompok keamanan
  - Hapus antarmuka
  - Memodifikasi antarmuka jaringan
- Untuk antarmuka jaringan
  - Buat tag
- Untuk antarmuka jaringan EC2 yang dikelola oleh Device Farm
  - Buat izin antarmuka jaringan

Kebijakan IAM lengkap berbunyi:

```
{
"Version": "2012-10-17",
"Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "ec2:DescribeNetworkInterfaces",
   "ec2:DescribeVpcs",
   "ec2:DescribeSubnets",
   "ec2:DescribeSecurityGroups"
  ],
  "Resource": "*"
 },
 {
  "Effect": "Allow",
  "Action": [
   "ec2:CreateNetworkInterface"
  ],
  "Resource": [
   "arn:aws:ec2:*:*:subnet/*",
   "arn:aws:ec2:*:*:security-group/*"
  ]
 },
 {
  "Effect": "Allow",
  "Action": [
   "ec2:CreateNetworkInterface"
  ],
  "Resource": [
   "arn:aws:ec2:*:*:network-interface/*"
  ],
  "Condition": {
   "StringEquals": {
    "aws:RequestTag/AWSDeviceFarmManaged": "true"
   }
  }
 },
 {
  "Effect": "Allow",
  "Action": [
   "ec2:CreateTags"
  ],
  "Resource": "arn:aws:ec2:*:*:network-interface/*",
```

```
    "Condition": {
     "StringEquals": {
      "ec2:CreateAction": "CreateNetworkInterface"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:CreateNetworkInterfacePermission",
     "ec2:DeleteNetworkInterface"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/AWSDeviceFarmManaged": "true"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": [
     "arn:aws:ec2:*:*:security-group/*",
     "arn:aws:ec2:*:*:instance/*"
    ]
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
     "StringEquals": {
      "aws:ResourceTag/AWSDeviceFarmManaged": "true"
     }
    }
   }
  ]
}
```

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi lebih lanjut, lihat Izin Peran Tertaut Layanan di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk Device Farm

Saat Anda menyediakan konfigurasi VPC untuk proyek pengujian seluler, Anda tidak perlu membuat peran terkait layanan secara manual. Saat Anda membuat sumber daya Device Farm pertama Anda diAWS Management Console, yangAWS CLI, atauAWSAPI, Device Farm membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat sumber daya Device Farm pertama Anda, Device Farm membuat peran terkait layanan untuk Anda lagi.

Anda juga dapat menggunakan konsol IAM untuk membuat peran terkait layanan denganPerangkat Pertaniankasus penggunaan. Di AWS CLI atau API AWS, buat peran yang terhubung dengan layanan dengan nama layanan `devicefarm.amazonaws.com`. Untuk informasi lebih lanjut, lihat Membuat Peran yang Terhubung dengan Layanan di Panduan Pengguna IAM. Jika Anda menghapus peran tertaut layanan ini, Anda dapat mengulang proses yang sama untuk membuat peran tersebut lagi.

Mengedit peran terkait layanan untuk Device Farm

Device Farm tidak memungkinkan Anda untuk mengedit AWSServiceRoleForDeviceFarm peran terkait layanan. Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi lebih lanjut, lihat Mengedit Peran Tertaut Layanan di Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Device Farm

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus

membersihkan sumber daya peran yang terhubung dengan layanan sebelum menghapusnya secara manual.

> **Note**
>
> Jika layanan Device Farm menggunakan peran saat Anda mencoba menghapus sumber daya, penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus peran terkait layanan secara manual menggunakan IAM

Gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran terkait layanan AWSServiceRoleForDeviceFarm. Untuk informasi lebih lanjut, lihat Menghapus Peran Tertaut Layanan di Panduan Pengguna IAM.

Wilayah yang Didukung untuk peran terkait layanan Device Farm

Device Farm mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat Wilayah dan titik akhir AWS.

Device Farm tidak mendukung penggunaan peran terkait layanan di setiap wilayah tempat layanan tersedia. Anda dapat menggunakan peran AWSServiceRoleForDeviceFarm di wilayah berikut.

Nama wilayah	Identitas wilayah	Dukungan di Device Farm
US East (Northern Virginia)	as-east-1	Tidak
Timur AS (Ohio)	us–east-2	Tidak
US West (Northern California)	us-west-1	Tidak
US West (Oregon)	us-west-2	Ya
Asia Pacific (Mumbai)	ap-south-1	Tidak
Asia Pacific (Osaka)	ap-northeast-3	Tidak
Asia Pacific (Seoul)	ap-northeast-2	Tidak

Nama wilayah	Identitas wilayah	Dukungan di Device Farm
Asia Pacific (Singapore)	ap-southeast-1	Tidak
Asia Pacific (Sydney)	ap-southeast-2	Tidak
Asia Pacific (Tokyo)	ap-northeast-1	Tidak
Canada (Central)	ca-central-1	Tidak
Eropa (Frankfurt)	eu-central-1	Tidak
Eropa (Irlandia)	eu-west-1	Tidak
Eropa (London)	eu-west-2	Tidak
Europe (Paris)	eu-west-3	Tidak
South America (São Paulo)	sa-east-1	Tidak
AWS GovCloud (US)	us-gov-west-1	Tidak

Prasyarat

Daftar berikut menjelaskan beberapa persyaratan dan saran untuk ditinjau saat membuat konfigurasi VPC-ENI:

- Perangkat pribadi harus ditetapkan ke perangkat AndaAWSAkun.
- Anda harus memilikiAWSpengguna akun atau peran dengan izin untuk membuat peran terkait Layanan. Saat menggunakan titik akhir Amazon VPC dengan fitur pengujian seluler Device Farm, Device Farm membuatAWS Identity and Access Management(IAM) peran terkait layanan.
- Device Farm dapat terhubung ke VPC hanya di`us-west-2`Wilayah. Jika Anda tidak memiliki VPC di`us-west-2`Wilayah, Anda perlu membuatnya. Kemudian, untuk mengakses sumber daya di VPC di Wilayah lain, Anda harus membuat koneksi peering antara VPC di`us-west-2`Wilayah dan VPC di Wilayah lain. Untuk informasi tentang mengintip VPC, lihatPanduan Peering Amazon VPC.

  Anda harus memverifikasi bahwa Anda memiliki akses ke VPC yang Anda tentukan saat Anda mengonfigurasi koneksi. Anda harus mengonfigurasi izin Amazon Elastic Compute Cloud (Amazon EC2) tertentu untuk Device Farm.

- Resolusi DNS diperlukan dalam VPC yang Anda gunakan.
- Setelah VPC Anda dibuat, Anda akan memerlukan informasi berikut tentang VPC di`us-west-2`Wilayah:
  - ID VPC
  - ID Subnet
  - ID grup keamanan
- Anda harus mengonfigurasi koneksi Amazon VPC berdasarkan per proyek. Pada saat ini, Anda hanya dapat mengonfigurasi satu konfigurasi VPC per proyek. Saat Anda mengonfigurasi VPC, Amazon VPC membuat antarmuka dalam VPC Anda dan menetapkannya ke subnet dan grup keamanan yang ditentukan. Semua sesi mendatang yang terkait dengan proyek akan menggunakan koneksi VPC yang dikonfigurasi.
- Anda tidak dapat menggunakan konfigurasi VPC-ENI bersama dengan fitur VPCE lama.
- Kami sangat merekomendasikantidak memperbarui proyek yang adadengan konfigurasi VPC-ENI karena proyek yang ada mungkin memiliki pengaturan VPCE yang bertahan pada level run. Sebaliknya, jika Anda sudah menggunakan fitur VPCE yang ada, gunakan VPC-ENI untuk semua proyek baru.

Menghubungkan ke Amazon VPC

Anda dapat mengonfigurasi dan memperbarui proyek Anda untuk menggunakan titik akhir Amazon VPC. Konfigurasi VPC-ENI dikonfigurasi berdasarkan per proyek. Sebuah proyek hanya dapat memiliki satu titik akhir VPC-ENI pada waktu tertentu. Untuk mengonfigurasi akses VPC untuk suatu proyek, Anda harus mengetahui detail berikut:

- ID VPC di`us-west-2`jika aplikasi Anda di-host di sana atau`us-west-2`ID VPC yang terhubung ke beberapa VPC lain di Wilayah yang berbeda.
- Grup keamanan yang berlaku untuk diterapkan pada koneksi.
- Subnet yang akan dikaitkan dengan koneksi. Ketika sesi dimulai, subnet terbesar yang tersedia digunakan. Kami merekomendasikan memiliki beberapa subnet yang terkait dengan zona ketersediaan yang berbeda untuk meningkatkan postur ketersediaan konektivitas VPC Anda.

Setelah Anda membuat konfigurasi VPC-ENI, Anda dapat memperbarui detailnya menggunakan konsol atau CLI menggunakan langkah-langkah di bawah ini.

Console

1. Masuk ke konsol Device Farm dihttps://console.aws.amazon.com/devicefarm.
2. Pada panel navigasi Device Farm, pilihPengujian Perangkat Seluler, lalu pilihProyek.
3. Di bawahProyek Pengujian Seluler, pilih nama proyek Anda dari daftar.
4. PilihPengaturan proyek.
5. DalamPengaturan Virtual Private Cloud (VPC)bagian, Anda dapat mengubahVPC,`Subnets`, dan`Security Groups`.
6. Pilih Save (Simpan).

CLI

Gunakan perintah AWS CLI berikut untuk memperbarui Amazon VPC:

```
$  aws devicefarm update-project \
--arn arn:aws:devicefarm:us-
west-2:111122223333:project:12345678-1111-2222-333-456789abcdef \
--vpc-config \
securityGroupIds=sg-02c1537701a7e3763,sg-005dadf9311efda25,\
subnetIds=subnet-09b1a45f9cac53717,subnet-09b1a45f9cac12345,\
vpcId=vpc-0238fb322af81a368
```

Anda juga dapat mengonfigurasi Amazon VPC saat membuat proyek Anda:

```
$  aws devicefarm create-project \
--name VPCDemo \
--vpc-config \
securityGroupIds=sg-02c1537701a7e3763,sg-005dadf9311efda25,\
subnetIds=subnet-09b1a45f9cac53717,subnet-09b1a45f9cac12345,\
vpcId=vpc-0238fb322af81a368
```

Batas

Batasan berikut berlaku untuk fitur VPC-ENI:

- Anda dapat menyediakan hingga lima grup keamanan dalam konfigurasi VPC proyek Device Farm.
- Anda dapat menyediakan hingga delapan subnet dalam konfigurasi VPC proyek Device Farm.

- Saat mengonfigurasi proyek Device Farm untuk bekerja dengan VPC Anda, subnet terkecil yang dapat Anda berikan harus memiliki minimal lima alamat IPv4 yang tersedia.
- Alamat IP publik tidak didukung saat ini. Sebagai gantinya, kami menyarankan Anda menggunakan subnet pribadi di proyek Device Farm Anda. Jika Anda membutuhkan akses internet publik selama pengujian, gunakangateway terjemahan alamat jaringan (NAT). Mengkonfigurasi proyek Device Farm dengan subnet publik tidak memberikan akses internet pengujian Anda atau alamat IP publik.
- Hanya lalu lintas keluar dari ENI yang dikelola layanan yang didukung. Ini berarti bahwa ENI tidak dapat menerima permintaan masuk yang tidak diminta dari VPC.

Mencatat panggilan AWS Device Farm API denganAWS CloudTrail

AWS Device Farm terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atauAWSlayanan di AWS Device Farm. CloudTrailmenangkap semua panggilan API untuk AWS Device Farm sebagai peristiwa. Panggilan yang diambil termasuk panggilan dari konsol AWS Device Farm dan panggilan kode ke operasi AWS Device Farm API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman berkelanjutanCloudTrailperistiwa ke bucket Amazon S3, termasuk peristiwa untuk AWS Device Farm. Jika Anda tidak membuat konfigurasi jejak, Anda masih dapat melihat kejadian terbaru dalam konsol CloudTrail di Riwayat peristiwa. Menggunakan informasi yang dikumpulkan olehCloudTrail, Anda dapat menentukan permintaan yang dibuat ke AWS Device Farm, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari lebih lanjut tentangCloudTrail, lihatAWS CloudTrailPanduan Pengguna.

Informasi AWS Device Farm diCloudTrail

CloudTrail diaktifkan pada akun AWS Anda saat Anda membuat akun tersebut. Saat aktivitas terjadi di AWS Device Farm, aktivitas tersebut direkam dalamCloudTrailacara bersama dengan lainnyaAWSacara layanan diRiwayat acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi lain, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan peristiwa yang sedang berlangsung diAWSakun, termasuk acara untuk AWS Device Farm, membuat jejak. Jejak memungkinkan CloudTrail untuk mengirim berkas log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat kejadian dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat membuat konfigurasi layanan AWS lainnya untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di log CloudTrail. Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak
- CloudTrailLayanan dan Integrasi yang Didukung
- Mengkonfigurasi Notifikasi Amazon SNS untukCloudTrail

- Menerima File Log CloudTrail dari Beberapa Wilayah dan Menerima File Log CloudTrail dari Beberapa Akun

KapanCloudTraillogging diaktifkan diAWSakun, panggilan API yang dilakukan ke tindakan Device Farm dilacak dalam file log. Catatan Device Farm ditulis bersama dengan yang lainAWScatatan layanan dalam file log. CloudTrail menentukan kapan membuat dan menulis ke berkas baru berdasarkan periode waktu dan ukuran berkas.

Semua tindakan Device Farm dicatat dan didokumentasikan diReferensi AWS CLIdanMengotomatisasi Perangkat Pertanian. Misalnya, panggilan untuk membuat proyek baru atau berjalan di Device Farm menghasilkan entri diCloudTrailfile log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Bahwa permintaan dibuat dengan kredensial pengguna root atau pengguna AWS Identity and Access Management (IAM).
- Bahwa permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Bahwa permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail.

Memahami entri file log AWS Device Farm

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang telah Anda tentukan. Berkas log CloudTrail berisi satu atau beberapa entri log. Peristuwa mewakili satu permintaan dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. Berkas log CloudTrail bukan jejak tumpukan terurut dari panggilan API publik, sehingga berkas tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkanCloudTrailentri log yang menunjukkan Device Farm`ListRuns`tindakan:

```
{
  "Records": [
    {
      "eventVersion": "1.03",
```

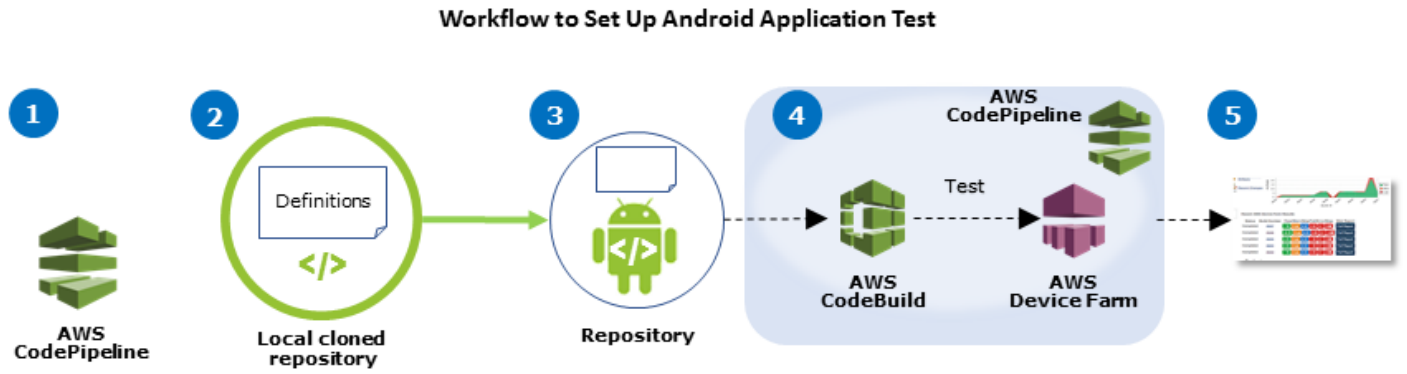
```
      "userIdentity": {
        "type": "Root",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::123456789012:root",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2015-07-08T21:13:35Z"
          }
        }
      },
      "eventTime":"2015-07-09T00:51:22Z",
      "eventSource": "devicefarm.amazonaws.com",
      "eventName":"ListRuns",
      "awsRegion":"us-west-2",
      "sourceIPAddress":"203.0.113.11",
      "userAgent":"example-user-agent-string",
      "requestParameters": {
        "arn":"arn:aws:devicefarm:us-west-2:123456789012:project:a9129b8c-
df6b-4cdd-8009-40a25EXAMPLE"},
        "responseElements": {
          "runs": [
            {
              "created": "Jul 8, 2015 11:26:12 PM",
              "name": "example.apk",
              "completedJobs": 2,
              "arn": "arn:aws:devicefarm:us-west-2:123456789012:run:a9129b8c-
df6b-4cdd-8009-40a256aEXAMPLE/1452d105-e354-4e53-99d8-6c993EXAMPLE",
              "counters": {
                "stopped": 0,
                "warned": 0,
                "failed": 0,
                "passed": 4,
                "skipped": 0,
                "total": 4,
                "errored": 0
              },
              "type": "BUILTIN_FUZZ",
              "status": "RUNNING",
              "totalJobs": 3,
              "platform": "ANDROID_APP",
              "result": "PENDING"
```

```
            },
            ... additional entries ...
          ]
        }
      }
    }
  ]
}
```

Menggunakan AWS Device Farm diCodePipelinetahap uji

Anda dapat menggunakanAWS CodePipelineuntuk menggabungkan pengujian aplikasi seluler yang dikonfigurasi di Device Farm ke dalam pipeline rilis otomatis yang dikelola AWS. Anda dapat mengonfigurasi pipeline untuk menjalankan pengujian sesuai permintaan, sesuai jadwal, atau sebagai bagian dari alur integrasi berkelanjutan.

Diagram berikut menunjukkan alur integrasi berkelanjutan di mana aplikasi Android dibangun dan diuji setiap kali push dilakukan ke repositorinya. Untuk membuat konfigurasi pipeline ini, lihatTutorial: Membangun dan Menguji Aplikasi Android Saat DorongGitHub.



1. Konfigurasikan	2. Tambahkan definisi	3. Dorong	4. Bangun dan uji	5. Laporkan
Konfigurasikan sumber daya pipa	Tambahkan definisi build dan test ke paket Anda	Dorong paket ke repositori Anda	Pembuatan aplikasi dan pengujian artefak keluaran build dimulai secara otomatis	Lihat hasil tes

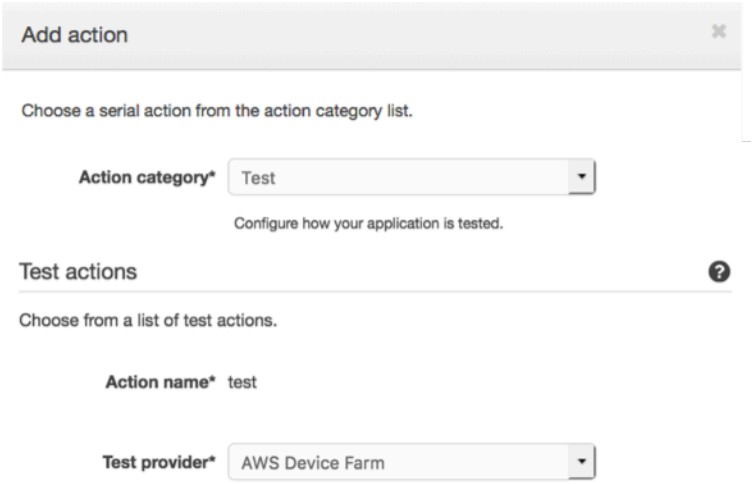
Untuk mempelajari cara mengonfigurasi pipeline yang terus-menerus menguji aplikasi yang dikompilasi (seperti iOS `.ipa`atau Android `.apk`file) sebagai sumbernya, lihatTutorial: Uji Aplikasi iOS Setiap Kali Anda Mengunggah File.ipa ke Bucket Amazon S3.

KonfigurasikanCodePipelineuntuk menggunakan pengujian Device Farm

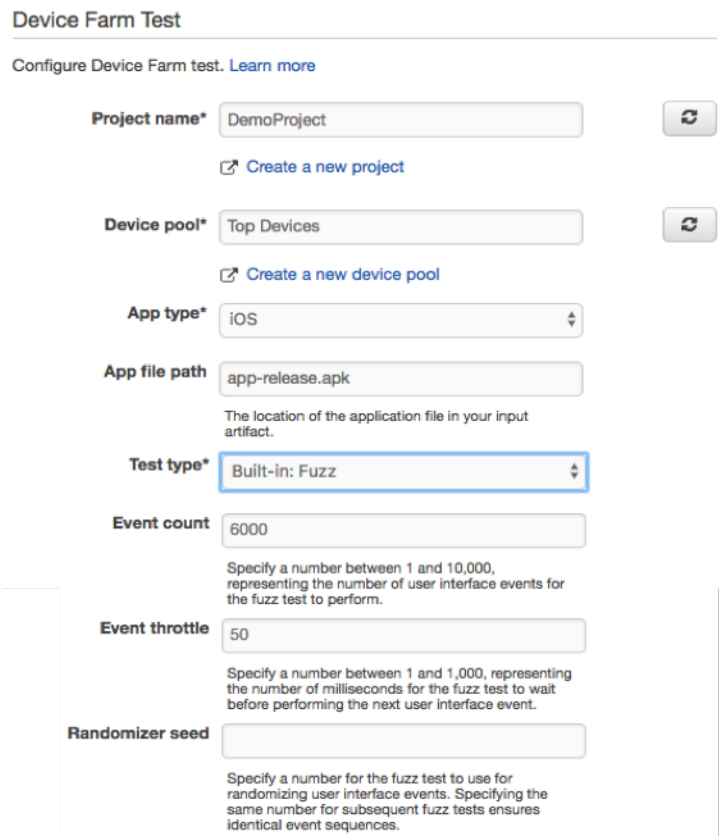
Dalam langkah-langkah ini, kami berasumsi bahwa Anda memilikimengkonfigurasi proyek Device Farmdanmembuat pipa. Pipa harus dikonfigurasi dengan tahap uji yang menerimaartefak masukanyang berisi definisi pengujian Anda dan file paket aplikasi yang dikompilasi. Artefak input tahap pengujian dapat berupa artefak keluaran dari sumber atau tahap build yang dikonfigurasi dalam pipeline Anda.

Untuk mengonfigurasi uji Device Farm yang dijalankan sebagaiCodePipelinetindakan uji

1. Masuk keAWS Management Consoledan bukaCodePipelinekonsol dihttps://console.aws.amazon.com/codepipeline/.
2. Pilih pipeline untuk rilis aplikasi Anda.
3. Pada panel tahap uji, pilih ikon pensil, lalu pilihAksi.
4. PadaTambahkan tindakanpanel, untukKategori aksi, pilihUji.
5. DiNama aksi, masukkan nama.
6. DiPenyedia tes, pilihPertanian Perangkat AWS.

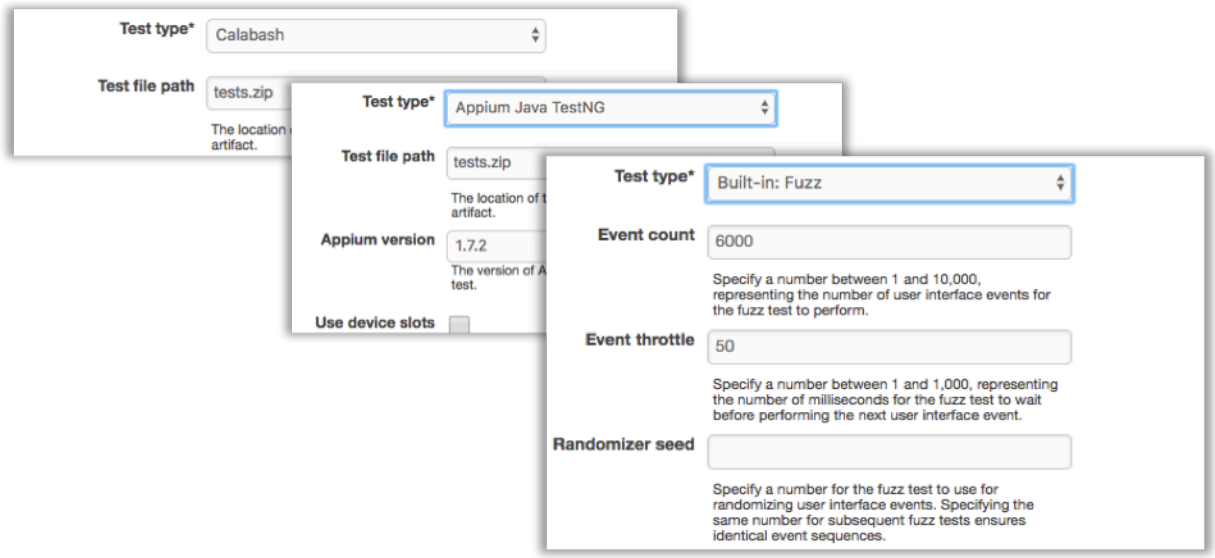


7. DiNama proyek, pilih proyek Device Farm yang ada atau pilihBuat proyek baru.
8. DiKolam perangkat, pilih kumpulan perangkat yang ada atau pilihBuat kumpulan perangkat baru. Jika Anda membuat kumpulan perangkat, Anda harus memilih satu set perangkat uji.
9. DiJenis aplikasi, pilih platform untuk aplikasi Anda.

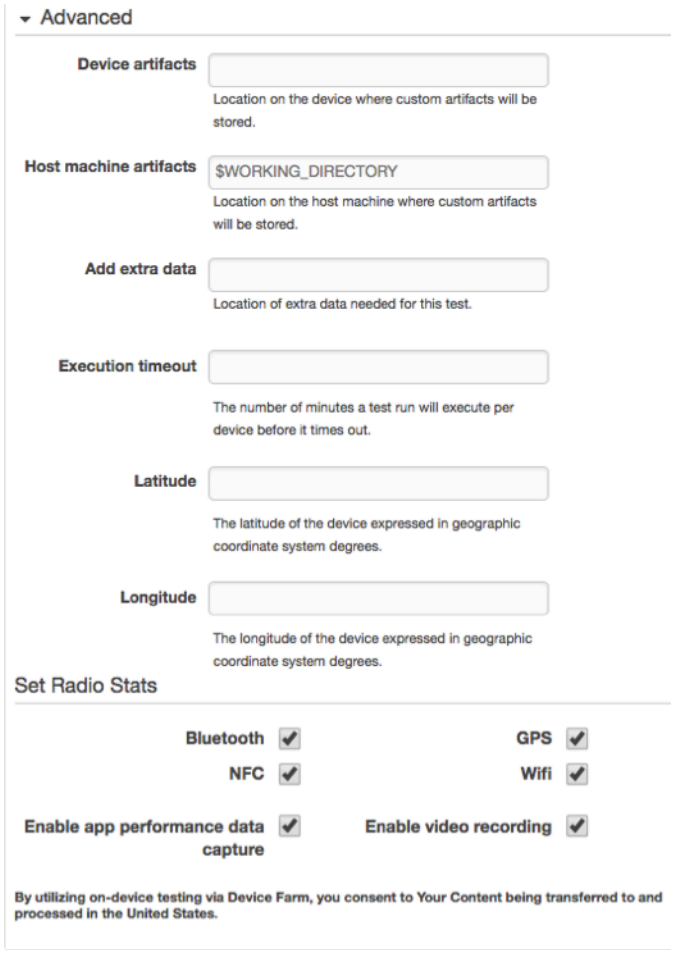


10. DiJalur file aplikasi, masukkan jalur paket aplikasi yang dikompilasi. Jalur relatif terhadap akar artefak input untuk pengujian Anda.
11. DiJenis uji, lakukan salah satu hal berikut:

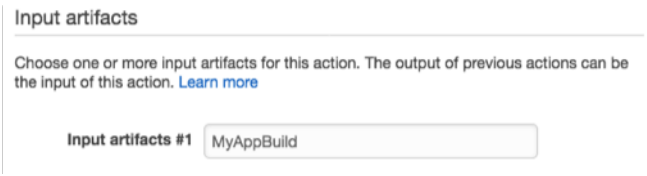
    - Jika Anda menggunakan salah satu pengujian Device Farm bawaan, pilih jenis pengujian yang dikonfigurasi dalam proyek Device Farm Anda.
    - Jika Anda tidak menggunakan salah satu pengujian bawaan Device Farm, diJalur file uji, masukkan jalur file definisi pengujian. Jalur relatif terhadap akar artefak input untuk pengujian Anda.



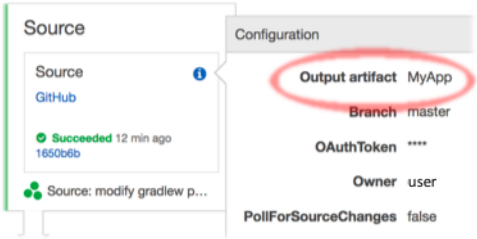
12. Di bidang yang tersisa, berikan konfigurasi yang sesuai untuk pengujian dan jenis aplikasi Anda.
13. (Opsional) DiLanjutan, berikan konfigurasi terperinci untuk uji coba Anda.



14. DiArtefak masukan, pilih artefak input yang cocok dengan artefak keluaran dari tahap yang datang sebelum tahap pengujian dalam pipa.



DiCodePipelinekonsol, Anda dapat menemukan nama artefak keluaran untuk setiap tahap dengan mengarahkan kursor ke ikon informasi dalam diagram pipa. Jika pipeline menguji aplikasi langsung dariSumberpanggung, pilihMyApp. Jika pipeline Anda menyertakanMembangunpanggung, pilihMyAppBuild.



15. Di bagian bawah panel, pilihTambahkan Tindakan.
16. DiCodePipelinepanel, pilihSimpan perubahan pipa, dan kemudian pilihSimpan perubahan.
17. Untuk mengirimkan perubahan dan memulai pembuatan pipeline, pilihRilis perubahan, dan kemudian pilihRilis.

AWS CLIreferensi untuk AWS Device Farm

Untuk menggunakanAWS Command Line Interface(AWS CLI) untuk menjalankan perintah Device Farm, lihatAWS CLIReferensi untuk AWS Device Farm.

Untuk informasi umum tentangAWS CLI, lihatAWS Command Line InterfacePanduan PenggunadanAWS CLIReferensi Perintah.

WindowsPowerShellreferensi untuk AWS Device Farm

Untuk menggunakan WindowsPowerShelluntuk menjalankan perintah Device Farm, lihatReferensi Perangkat Pertanian CmdletdiAWS Tools for Windows PowerShellCmdlet Referensi. Untuk informasi lebih lanjut, lihatMenyiapkan AWS Tools untuk WindowsPowerShelldiAWS Tools for Windows PowerShellPanduan Pengguna.

Mengotomatiskan AWS Device Farm

Akses terprogram ke Device Farm adalah cara ampuh untuk mengotomatiskan tugas-tugas umum yang perlu Anda selesaikan, seperti menjadwalkan proses atau mengunduh artefak untuk dijalankan, suite, atau pengujian. TheAWSSDK danAWS CLImenyediakan sarana untuk melakukannya.

TheAWSSDK menyediakan akses ke setiapAWSlayanan, termasuk Device Farm, Amazon S3, dan banyak lagi. Untuk informasi selengkapnya, lihat

- sangAWSalat dan SDK
- sangReferensi API AWS Device Farm

Contoh: MenggunakanAWSSDK untuk memulai menjalankan Device Farm dan mengumpulkan artefak

Contoh berikut memberikan demonstrasi awal-ke-akhir tentang bagaimana Anda dapat menggunakanAWSSDK untuk bekerja dengan Device Farm. Contoh ini melakukan hal berikut:

- Mengunggah paket pengujian dan aplikasi ke Device Farm
- Memulai uji coba dan menunggu penyelesaiannya (atau kegagalan)
- Mengunduh semua artefak yang diproduksi oleh suite uji

Contoh ini tergantung pada pihak ketiga`requests`paket untuk berinteraksi dengan HTTP.

```
import boto3
import os
import requests
import string
import random
import time
import datetime
import time
import json

# The following script runs a test through Device Farm
#
# Things you have to change:
config = {
```

```
    # This is our app under test.
    "appFilePath":"app-debug.apk",
    "projectArn": "arn:aws:devicefarm:us-
west-2:111122223333:project:1b99bcff-1111-2222-ab2f-8c3c733c55ed",
    # Since we care about the most popular devices, we'll use a curated pool.
    "testSpecArn":"arn:aws:devicefarm:us-west-2::upload:101e31e8-12ac-11e9-ab14-
d663bd873e83",
    "poolArn":"arn:aws:devicefarm:us-west-2::devicepool:082d10e5-d7d7-48a5-ba5c-
b33d66efa1f5",
    "namePrefix":"MyAppTest",
    # This is our test package. This tutorial won't go into how to make these.
    "testPackage":"tests.zip"
}

client = boto3.client('devicefarm')

unique =
 config['namePrefix']+"-"+(datetime.date.today().isoformat())+(''.join(random.sample(string.asc

print(f"The unique identifier for this run is going to be {unique} -- all uploads will
 be prefixed with this.")

def upload_df_file(filename, type_, mime='application/octet-stream'):
    response = client.create_upload(projectArn=config['projectArn'],
        name = (unique)+"_"+os.path.basename(filename),
        type=type_,
        contentType=mime
        )
    # Get the upload ARN, which we'll return later.
    upload_arn = response['upload']['arn']
    # We're going to extract the URL of the upload and use Requests to upload it
    upload_url = response['upload']['url']
    with open(filename, 'rb') as file_stream:
        print(f"Uploading {filename} to Device Farm as {response['upload']['name']}...
 ",end='')
        put_req = requests.put(upload_url, data=file_stream, headers={"content-
type":mime})
        print(' done')
        if not put_req.ok:
            raise Exception("Couldn't upload, requests said we're not ok. Requests
 says: "+put_req.reason)
    started = datetime.datetime.now()
    while True:
```

```
        print(f"Upload of {filename} in state {response['upload']['status']} after
 "+str(datetime.datetime.now() - started))
        if response['upload']['status'] == 'FAILED':
            raise Exception("The upload failed processing. DeviceFarm says reason
 is: \n"+(response['upload']['message'] if 'message' in response['upload'] else
 response['upload']['metadata']))
        if response['upload']['status'] == 'SUCCEEDED':
            break
        time.sleep(5)
        response = client.get_upload(arn=upload_arn)
    print("")
    return upload_arn

our_upload_arn = upload_df_file(config['appFilePath'], "ANDROID_APP")
our_test_package_arn = upload_df_file(config['testPackage'],
 'APPIUM_PYTHON_TEST_PACKAGE')
print(our_upload_arn, our_test_package_arn)
# Now that we have those out of the way, we can start the test run...
response = client.schedule_run(
    projectArn = config["projectArn"],
    appArn = our_upload_arn,
    devicePoolArn = config["poolArn"],
    name=unique,
    test = {
        "type":"APPIUM_PYTHON",
        "testSpecArn": config["testSpecArn"],
        "testPackageArn": our_test_package_arn
        }
    )
run_arn = response['run']['arn']
start_time = datetime.datetime.now()
print(f"Run {unique} is scheduled as arn {run_arn} ")

try:

    while True:
        response = client.get_run(arn=run_arn)
        state = response['run']['status']
        if state == 'COMPLETED' or state == 'ERRORED':
            break
        else:
            print(f" Run {unique} in state {state}, total time
 "+str(datetime.datetime.now()-start_time))
            time.sleep(10)
```

```
except:
    # If something goes wrong in this process, we stop the run and exit.

    client.stop_run(arn=run_arn)
    exit(1)
print(f"Tests finished in state {state} after "+str(datetime.datetime.now() -
 start_time))
# now, we pull all the logs.
jobs_response = client.list_jobs(arn=run_arn)
# Save the output somewhere. We're using the unique value, but you could use something
 else
save_path = os.path.join(os.getcwd(), unique)
os.mkdir(save_path)
# Save the last run information
for job in jobs_response['jobs'] :
    # Make a directory for our information
    job_name = job['name']
    os.makedirs(os.path.join(save_path, job_name), exist_ok=True)
    # Get each suite within the job
    suites = client.list_suites(arn=job['arn'])['suites']
    for suite in suites:
        for test in client.list_tests(arn=suite['arn'])['tests']:
            # Get the artifacts
            for artifact_type in ['FILE','SCREENSHOT','LOG']:
                artifacts = client.list_artifacts(
                    type=artifact_type,
                    arn = test['arn']
                )['artifacts']
                for artifact in artifacts:
                    # We replace : because it has a special meaning in Windows & macos
                    path_to = os.path.join(save_path, job_name, suite['name'],
 test['name'].replace(':','_') )
                    os.makedirs(path_to, exist_ok=True)
                    filename =
 artifact['type']+"_"+artifact['name']+"."+artifact['extension']
                    artifact_save_path = os.path.join(path_to, filename)
                    print("Downloading "+artifact_save_path)
                    with open(artifact_save_path, 'wb') as fn,
 requests.get(artifact['url'],allow_redirects=True) as request:
                        fn.write(request.content)
                    #/for artifact in artifacts
                #/for artifact type in []
            #/ for test in ()[]
        #/ for suite in suites
```

```
    #/ for job in _[]
# done
print("Finished")
```

Memecahkan masalah kesalahan Device Farm

Di bagian ini, Anda akan menemukan pesan kesalahan dan prosedur untuk membantu Anda memperbaiki masalah umum dengan Device Farm.

Topik

Memecahkan masalah pengujian aplikasi Android di AWS Device Farm

Topik berikut mencantumkan pesan galat yang terjadi selama pengunggahan pengujian aplikasi Android dan merekomendasikan solusi untuk mengatasi setiap kesalahan.

> **Note**
>
> Petunjuk di bawah ini didasarkan pada Linux x86_64 dan Mac.

ANDROID_APP_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka aplikasi Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat unzip paket aplikasi tanpa kesalahan. Dalam contoh berikut, nama paket adalahapp-debug.apk.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip app-debug.apk
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket aplikasi Android yang valid harus menghasilkan output seperti berikut:

   ```
   .
   |-- AndroidManifest.xml
   |-- classes.dex
   |-- resources.arsc
   |-- assets (directory)
   |-- res (directory)
   `-- META-INF (directory)
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian Android di AWS Device Farm.

ANDROID_APP_AAPT_DEBUG_BADGING_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat mengekstrak informasi tentang aplikasi Anda. Harap verifikasi bahwa aplikasi tersebut valid dengan menjalankan perintah`aapt debug badging <path to your test package>`, dan coba lagi setelah perintah tidak mencetak kesalahan apa pun.

Selama proses validasi unggahan, AWS Device Farm mem-parsing informasi dari output`aapt debug badging <path to your package>`perintah.

Pastikan Anda dapat menjalankan perintah ini di aplikasi Android Anda dengan sukses. Dalam contoh berikut, nama paket adalahapp-debug.apk.

- Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah:

  ```
  $ aapt debug badging app-debug.apk
  ```

  Paket aplikasi Android yang valid harus menghasilkan output seperti berikut:

  ```
  package: name='com.amazon.aws.adf.android.referenceapp' versionCode='1'
   versionName='1.0' platformBuildVersionName='5.1.1-1819727'
  sdkVersion:'9'
  application-label:'ReferenceApp'
  application: label='ReferenceApp' icon='res/mipmap-mdpi-v4/ic_launcher.png'
  application-debuggable
  launchable-activity:
   name='com.amazon.aws.adf.android.referenceapp.Activities.MainActivity'
   label='ReferenceApp' icon=''
  uses-feature: name='android.hardware.bluetooth'
  uses-implied-feature: name='android.hardware.bluetooth' reason='requested
   android.permission.BLUETOOTH permission, and targetSdkVersion > 4'
  main
  supports-screens: 'small' 'normal' 'large' 'xlarge'
  supports-any-density: 'true'
  locales: '--_--'
  densities: '160' '213' '240' '320' '480' '640'
  ```

  Untuk informasi selengkapnya, lihat Bekerja dengan pengujian Android di AWS Device Farm.

ANDROID_APP_PACKAGE_NAME_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai nama paket dalam aplikasi Anda. Harap verifikasi bahwa aplikasi tersebut valid dengan menjalankan perintah`aapt debug badging <path to your test package>`, dan coba lagi setelah menemukan nilai nama paket di belakang kata kunci “package: name.”

Selama proses validasi unggahan, AWS Device Farm mem-parsing nilai nama paket dari output`aapt debug badging <path to your package>`perintah.

Pastikan Anda dapat menjalankan perintah ini di aplikasi Android Anda dan menemukan nilai nama paket dengan sukses. Dalam contoh berikut, nama paket adalahapp-debug.apk.

- Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

  ```
  $ aapt debug badging app-debug.apk | grep "package: name="
  ```

  Paket aplikasi Android yang valid harus menghasilkan output seperti berikut:

  ```
  package: name='com.amazon.aws.adf.android.referenceapp' versionCode='1'
   versionName='1.0' platformBuildVersionName='5.1.1-1819727'
  ```

  Untuk informasi selengkapnya, lihat Bekerja dengan pengujian Android di AWS Device Farm.

ANDROID_APP_SDK_VERSION_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai versi SDK di aplikasi Anda. Harap verifikasi bahwa aplikasi tersebut valid dengan menjalankan perintah`aapt debug badging <path to`

your test package>, dan coba lagi setelah menemukan nilai versi SDK di belakang kata kuncisdkVersion.

Selama proses validasi unggahan, AWS Device Farm mem-parsing nilai versi SDK dari outputaapt debug badging *<path to your package>*perintah.

Pastikan Anda dapat menjalankan perintah ini di aplikasi Android Anda dan menemukan nilai nama paket dengan sukses. Dalam contoh berikut, nama paket adalahapp-debug.apk.

- Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

  ```
  $ aapt debug badging app-debug.apk | grep "sdkVersion"
  ```

  Paket aplikasi Android yang valid harus menghasilkan output seperti berikut:

  ```
  sdkVersion:'9'
  ```

  Untuk informasi selengkapnya, lihat Bekerja dengan pengujian Android di AWS Device Farm.

ANDROID_APP_AAPT_DUMP_XMLTREE_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan yang validAndroidManifest.xml. dalam aplikasi Anda. Harap verifikasi bahwa paket pengujian valid dengan menjalankan perintahaapt dump xmltree *<path to your test package>* AndroidManifest.xml, dan coba lagi setelah perintah tidak mencetak kesalahan apa pun.

Selama proses validasi unggahan, AWS Device Farm mem-parsing informasi dari pohon parse XMLuntuk file XMLyang terdapat dalam paket menggunakan perintahaapt dump xmltree *<path to your package>* AndroidManifest.xml.

Pastikan Anda dapat menjalankan perintah ini di aplikasi Android Anda dengan sukses. Dalam contoh berikut, nama paket adalahapp-debug.apk.

- Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ aapt dump xmltree app-debug.apk. AndroidManifest.xml
```

Paket aplikasi Android yang valid harus menghasilkan output seperti berikut:

```
N: android=http://schemas.android.com/apk/res/android
  E: manifest (line=2)
    A: android:versionCode(0x0101021b)=(type 0x10)0x1
    A: android:versionName(0x0101021c)="1.0" (Raw: "1.0")
    A: package="com.amazon.aws.adf.android.referenceapp" (Raw:
 "com.amazon.aws.adf.android.referenceapp")
    A: platformBuildVersionCode=(type 0x10)0x16 (Raw: "22")
    A: platformBuildVersionName="5.1.1-1819727" (Raw: "5.1.1-1819727")
    E: uses-sdk (line=7)
      A: android:minSdkVersion(0x0101020c)=(type 0x10)0x9
      A: android:targetSdkVersion(0x01010270)=(type 0x10)0x16
    E: uses-permission (line=11)
      A: android:name(0x01010003)="android.permission.INTERNET" (Raw:
 "android.permission.INTERNET")
    E: uses-permission (line=12)
      A: android:name(0x01010003)="android.permission.CAMERA" (Raw:
 "android.permission.CAMERA")
```

Untuk informasi selengkapnya, lihat Bekerja dengan pengujian Android di AWS Device Farm.

ANDROID_APP_DEVICE_ADMIN_PERMISSIONS

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan bahwa aplikasi Anda memerlukan izin admin perangkat. Harap verifikasi bahwa izin tidak diperlukan dengan menjalankan perintah`aapt dump xmltree` *`<path to your test package>`* `AndroidManifest.xml`, dan coba lagi setelah memastikan bahwa output tidak mengandung kata kunci`android.permission.BIND_DEVICE_ADMIN`.

Selama proses validasi unggahan, AWS Device Farm mem-parsing informasi izin dari pohon parse xmluntuk file xmlyang terdapat dalam paket menggunakan perintah`aapt dump xmltree <path to your package> AndroidManifest.xml`.

Pastikan aplikasi Anda tidak memerlukan izin admin perangkat. Dalam contoh berikut, nama paket adalahapp-debug.apk.

- Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ aapt dump xmltree app-debug.apk AndroidManifest.xml
```

Anda harus menemukan output seperti berikut:

```
N: android=http://schemas.android.com/apk/res/android
  E: manifest (line=2)
    A: android:versionCode(0x0101021b)=(type 0x10)0x1
    A: android:versionName(0x0101021c)="1.0" (Raw: "1.0")
    A: package="com.amazonaws.devicefarm.android.referenceapp" (Raw:
 "com.amazonaws.devicefarm.android.referenceapp")
    A: platformBuildVersionCode=(type 0x10)0x16 (Raw: "22")
    A: platformBuildVersionName="5.1.1-1819727" (Raw: "5.1.1-1819727")
    E: uses-sdk (line=7)
      A: android:minSdkVersion(0x0101020c)=(type 0x10)0xa
      A: android:targetSdkVersion(0x01010270)=(type 0x10)0x16
    E: uses-permission (line=11)
      A: android:name(0x01010003)="android.permission.INTERNET" (Raw:
 "android.permission.INTERNET")
    E: uses-permission (line=12)
      A: android:name(0x01010003)="android.permission.CAMERA" (Raw:
 "android.permission.CAMERA")
        ……
```

Jika aplikasi Android valid, output tidak boleh berisi yang berikut:`A: android:name(0x01010003)="android.permission.BIND_DEVICE_ADMIN" (Raw: "android.permission.BIND_DEVICE_ADMIN")`.

Untuk informasi selengkapnya, lihat Bekerja dengan pengujian Android di AWS Device Farm.

Jendela tertentu di aplikasi Android saya menampilkan layar kosong atau hitam

Jika Anda menguji aplikasi Android dan melihat bahwa jendela tertentu dalam aplikasi muncul dengan layar hitam dalam perekaman video Device Farm dari pengujian Anda, aplikasi Anda mungkin menggunakan Android`FLAG_SECURE`fitur. Bendera ini (seperti yang dijelaskan dalamDokumentasi resmi Android) digunakan untuk mencegah jendela tertentu dari suatu aplikasi direkam oleh alat perekam layar. Akibatnya, fitur perekaman layar Device Farm (untuk otomatisasi dan pengujian akses jarak jauh) dapat menampilkan layar hitam sebagai pengganti jendela aplikasi Anda jika jendela menggunakan bendera ini.

Bendera ini sering digunakan oleh pengembang untuk halaman dalam aplikasi mereka yang berisi informasi sensitif seperti halaman login. Jika Anda melihat layar hitam di tempat layar aplikasi Anda untuk halaman tertentu seperti halaman loginnya, bekerja dengan pengembang Anda untuk mendapatkan build aplikasi yang tidak menggunakan bendera ini untuk pengujian.

Selain itu, perhatikan bahwa Device Farm masih dapat berinteraksi dengan jendela aplikasi yang memiliki bendera ini. Jadi, jika halaman login aplikasi Anda muncul sebagai layar hitam, Anda mungkin masih dapat memasukkan kredensi Anda untuk masuk ke aplikasi (dan dengan demikian melihat halaman yang tidak diblokir oleh`FLAG_SECURE`bendera).

Memecahkan masalah pengujian Appium Java JUnit di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian JUnit Appium Java dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

> **Note**
>
> Petunjuk di bawah ini didasarkan pada Linux x86_64 dan Mac.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket Appium Java JUnit yang valid harus menghasilkan output seperti berikut:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
        |— com.some-dependency.bar-4.1.jar
        |— com.another-dependency.thing-1.0.jar
        |— joda-time-2.7.jar
        `— log4j-1.2.14.jar
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_DEPENDENCY_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori dependency-jars di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori dependency-jars ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan*toples ketergantungan*direktori di dalam direktori kerja:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |— com.some-dependency.bar-4.1.jar
         |— com.another-dependency.thing-1.0.jar
         |— joda-time-2.7.jar
         `— log4j-1.2.14.jar
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_JAR_MISSING_IN_DEPENDENCY_DI

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file JAR di pohon direktori dependency-jars. Harap unzip paket pengujian Anda dan kemudian buka direktori dependency-jars, verifikasi bahwa setidaknya satu file JAR ada di direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*berkas di dalam*toples ketergantungan*direktori:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |— com.some-dependency.bar-4.1.jar
         |— com.another-dependency.thing-1.0.jar
         |— joda-time-2.7.jar
         `— log4j-1.2.14.jar
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_TESTS_JAR_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file*-tests.jar dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa setidaknya satu file*-tests.jar ada dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*file seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |– zip-with-dependencies.zip (this .zip file contains all of the items)
   `– dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |– com.some-dependency.bar-4.1.jar
         |– com.another-dependency.thing-1.0.jar
         |– joda-time-2.7.jar
         `– log4j-1.2.14.jar
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_CLASS_FILE_MISSING_IN_TESTS_JA

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file kelas dalam file JAR tes. Harap unzip paket pengujian Anda dan kemudian unjar file JAR tes, verifikasi bahwa setidaknya satu file kelas ada di dalam file JAR, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan setidaknya satu file jar seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |– zip-with-dependencies.zip (this .zip file contains all of the items)
   `– dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |– com.some-dependency.bar-4.1.jar
         |– com.another-dependency.thing-1.0.jar
         |– joda-time-2.7.jar
         `– log4j-1.2.14.jar
   ```

3. Setelah Anda berhasil mengekstrak file, Anda harus menemukan setidaknya satu kelas di pohon direktori kerja dengan menjalankan perintah:

```
$ tree .
```

Anda akan melihat output seperti ini:

```
.
|— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
everything built from the ./src/test directory)
|- one-class-file.class
|- folder
|    `-another-class-file.class
|— zip-with-dependencies.zip (this .zip file contains all of the items)
`— dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_JUNIT_VERSION_VALUE_UNKNOWN

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai versi JUnit. Harap unzip paket pengujian Anda dan buka direktori dependency-jars, verifikasi bahwa file JUnit JAR ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip zip-with-dependencies.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
tree .
```

Outputnya akan terlihat seperti ini:

```
.
|— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|— zip-with-dependencies.zip (this .zip file contains all of the items)
`— dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
    |— junit-4.10.jar
    |— com.some-dependency.bar-4.1.jar
    |— com.another-dependency.thing-1.0.jar
    |— joda-time-2.7.jar
    `— log4j-1.2.14.jar
```

Jika paket Appium Java JUnit valid, Anda akan menemukan file dependensi JUnit yang mirip dengan file jar*junit-4.10.jar*dalam contoh kita. Nama harus terdiri dari kata kunci*junit*dan nomor versinya, yang dalam contoh ini adalah 4.10.

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_JUNIT_TEST_PACKAGE_INVALID_JUNIT_VERSION

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan versi JUnit lebih rendah dari versi minimum 4.10 yang kami dukung. Harap ubah versi JUnit dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip zip-with-dependencies.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan file ketergantungan JUnit seperti*junit-4.10.jar*dalam contoh kita dan nomor versinya, yang dalam contoh kita adalah 4.10:

```
.
|— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|— zip-with-dependencies.zip (this .zip file contains all of the items)
`— dependency-jars (this is the directory that contains all of your dependencies,
 built as JAR files)
      |— junit-4.10.jar
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

> **Note**
>
> Pengujian Anda mungkin tidak dijalankan dengan benar jika versi JUnit yang ditentukan dalam paket pengujian Anda lebih rendah dari versi minimum 4.10 yang kami dukung.

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

Memecahkan masalah pengujian aplikasi web Appium Java JUnit di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian aplikasi Appium JUnit JUnit Web dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan. Untuk informasi selengkapnya tentang penggunaan Appium dengan Device Farm, lihatthe section called “Appium”.

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket Appium Java JUnit yang valid harus menghasilkan output seperti berikut:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   ```

```
`— dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_DEPENDENCY_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> ⚠ Warning
>
> Kami tidak dapat menemukan direktori dependency-jars di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori dependency-jars ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan*toples ketergantungan*direktori di dalam direktori kerja:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
   ```

```
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_JAR_MISSING_IN_DEPENDEN

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file JAR di pohon direktori dependency-jars. Harap unzip paket pengujian Anda dan kemudian buka direktori dependency-jars, verifikasi bahwa setidaknya satu file JAR ada di direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*berkas di dalam*toples ketergantungan*direktori:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |— com.some-dependency.bar-4.1.jar
         |— com.another-dependency.thing-1.0.jar
   ```

```
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_TESTS_JAR_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file*-tests.jar dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa setidaknya satu file*-tests.jar ada dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*file seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |– zip-with-dependencies.zip (this .zip file contains all of the items)
   `– dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |– com.some-dependency.bar-4.1.jar
         |– com.another-dependency.thing-1.0.jar
   ```

```
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_CLASS_FILE_MISSING_IN_TES

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file kelas dalam file JAR tes. Harap unzip paket pengujian Anda dan kemudian unjar file JAR tes, verifikasi bahwa setidaknya satu file kelas ada di dalam file JAR, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan setidaknya satu file jar seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |– zip-with-dependencies.zip (this .zip file contains all of the items)
   `– dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |– com.some-dependency.bar-4.1.jar
         |– com.another-dependency.thing-1.0.jar
   ```

```
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

3. Setelah Anda berhasil mengekstrak file, Anda harus menemukan setidaknya satu kelas di pohon direktori kerja dengan menjalankan perintah:

```
$ tree .
```

Anda akan melihat output seperti ini:

```
.
|— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
everything built from the ./src/test directory)
|- one-class-file.class
|- folder
|    `-another-class-file.class
|— zip-with-dependencies.zip (this .zip file contains all of the items)
`— dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_JUNIT_VERSION_VALUE_UNK

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai versi JUnit. Harap unzip paket pengujian Anda dan buka direktori dependency-jars, verifikasi bahwa file JUnit JAR ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip zip-with-dependencies.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
tree .
```

Outputnya akan terlihat seperti ini:

```
.
|— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|— zip-with-dependencies.zip (this .zip file contains all of the items)
`— dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |— junit-4.10.jar
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

Jika paket Appium Java JUnit valid, Anda akan menemukan file dependensi JUnit yang mirip dengan file jar*junit-4.10.jar*dalam contoh kita. Nama harus terdiri dari kata kunci*junit*dan nomor versinya, yang dalam contoh ini adalah 4.10.

APPIUM_WEB_JAVA_JUNIT_TEST_PACKAGE_INVALID_JUNIT_VERSION

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan versi JUnit lebih rendah dari versi minimum 4.10 yang kami dukung. Harap ubah versi JUnit dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip zip-with-dependencies.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan file ketergantungan JUnit seperti*junit-4.10.jar*dalam contoh kita dan nomor versinya, yang dalam contoh kita adalah 4.10:

```
.
|— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|— zip-with-dependencies.zip (this .zip file contains all of the items)
`— dependency-jars (this is the directory that contains all of your dependencies,
 built as JAR files)
      |— junit-4.10.jar
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

> **Note**
>
> Pengujian Anda mungkin tidak dijalankan dengan benar jika versi JUnit yang ditentukan dalam paket pengujian Anda lebih rendah dari versi minimum 4.10 yang kami dukung.

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

Memecahkan masalah pengujian Appium Java TestNG di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian Appium Java TestNG dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

> **Note**
>
> Petunjuk di bawah ini didasarkan pada Linux x86_64 dan Mac.

APPIUM_JAVA_TESTNG_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket Appium Java JUnit yang valid harus menghasilkan output seperti berikut:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   ```

```
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_TESTNG_TEST_PACKAGE_DEPENDENCY_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan`dependency-jars`direktori di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa`dependency-jars`direktori ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan*`toples ketergantungan`*direktori di dalam direktori kerja.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   ```

```
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_TESTNG_TEST_PACKAGE_JAR_MISSING_IN_DEPENDENCY_

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file JAR di pohon direktori dependency-jars. Harap unzip paket pengujian Anda dan kemudian buka direktori dependency-jars, verifikasi bahwa setidaknya satu file JAR ada di direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*berkas di dalam*toples ketergantungan*direktori.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   ```

```
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_TESTNG_TEST_PACKAGE_TESTS_JAR_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file*-tests.jar dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa setidaknya satu file*-tests.jar ada dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*file seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

   ```
   .
   ```

```
|– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_JAVA_TESTNG_TEST_PACKAGE_CLASS_FILE_MISSING_IN_TESTS

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file kelas dalam file JAR tes. Harap unzip paket pengujian Anda dan kemudian unjar file JAR tes, verifikasi bahwa setidaknya satu file kelas ada di dalam file JAR, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan setidaknya satu file jar seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

```
.
|– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
 everything built from the ./src/test directory)
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

3. Untuk mengekstrak file dari file jar, Anda dapat menjalankan perintah berikut:

```
$ jar xf acme-android-appium-1.0-SNAPSHOT-tests.jar
```

4. Setelah Anda berhasil mengekstrak file, jalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan setidaknya satu kelas di pohon direktori kerja:

```
.
|– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
everything built from the ./src/test directory)
|- one-class-file.class
|- folder
|    `– another-class-file.class
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

Memecahkan masalah aplikasi web Appium Java TestNG di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian aplikasi Web Appium Java TestNG dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

APPIUM_WEB_JAVA_TESTNG_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket Appium Java JUnit yang valid harus menghasilkan output seperti berikut:

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |— com.some-dependency.bar-4.1.jar
         |— com.another-dependency.thing-1.0.jar
   ```

```
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_JAVA_TESTNG_TEST_PACKAGE_DEPENDENCY_DIR_MISSIN

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori dependency-jars di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori dependency-jars ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan*toples ketergantungan*direktori di dalam direktori kerja.

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
        |— com.some-dependency.bar-4.1.jar
        |— com.another-dependency.thing-1.0.jar
   ```

```
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_JAVA_TESTNG_TEST_PACKAGE_JAR_MISSING_IN_DEPENDE

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file JAR di pohon direktori dependency-jars. Harap unzip paket pengujian Anda dan kemudian buka direktori dependency-jars, verifikasi bahwa setidaknya satu file JAR ada di direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*berkas di dalam*toples ketergantungan*direktori.

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   `— dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
         |— com.some-dependency.bar-4.1.jar
         |— com.another-dependency.thing-1.0.jar
   ```

```
    |– joda-time-2.7.jar
    `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_JAVA_TESTNG_TEST_PACKAGE_TESTS_JAR_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file*-tests.jar dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa setidaknya satu file*-tests.jar ada dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Java JUnit valid, Anda akan menemukan setidaknya satu*toples*file seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*–tests.jar*.

   ```
   .
   |– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |– zip-with-dependencies.zip (this .zip file contains all of the items)
   `– dependency-jars  (this is the directory that contains all of your dependencies,
    built as JAR files)
   ```

```
      |— com.some-dependency.bar-4.1.jar
      |— com.another-dependency.thing-1.0.jar
      |— joda-time-2.7.jar
      `— log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_JAVA_TESTNG_TEST_PACKAGE_CLASS_FILE_MISSING_IN_T

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file kelas dalam file JAR tes. Harap unzip paket pengujian Anda dan kemudian unjar file JAR tes, verifikasi bahwa setidaknya satu file kelas ada di dalam file JAR, dan coba lagi.

Dalam contoh berikut, nama paket adalahzip-with-dependencies.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip zip-with-dependencies.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan setidaknya satu file jar seperti*acme-android-appium-1.0-snapshot-tests.jar*dalam contoh kita. Nama file mungkin berbeda, tetapi harus diakhiri dengan*—tests.jar*.

   ```
   .
   |— acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
    built from the ./src/main directory)
   |— acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
    everything built from the ./src/test directory)
   |— zip-with-dependencies.zip (this .zip file contains all of the items)
   ```

```
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

3. Untuk mengekstrak file dari file jar, Anda dapat menjalankan perintah berikut:

```
$ jar xf acme-android-appium-1.0-SNAPSHOT-tests.jar
```

4. Setelah Anda berhasil mengekstrak file, jalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan setidaknya satu kelas di pohon direktori kerja:

```
.
|– acme-android-appium-1.0-SNAPSHOT.jar (this is the JAR containing everything
 built from the ./src/main directory)
|– acme-android-appium-1.0-SNAPSHOT-tests.jar (this is the JAR containing
everything built from the ./src/test directory)
|- one-class-file.class
|- folder
|    `– another-class-file.class
|– zip-with-dependencies.zip (this .zip file contains all of the items)
`– dependency-jars  (this is the directory that contains all of your dependencies,
 built as JAR files)
      |– com.some-dependency.bar-4.1.jar
      |– com.another-dependency.thing-1.0.jar
      |– joda-time-2.7.jar
      `– log4j-1.2.14.jar
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

Memecahkan masalah pengujian Appium Python di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian Appium Python dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

APPIUM_PYTHON_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP uji Appium Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket Appium Python yang valid harus menghasilkan output seperti berikut:

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `-- test_unittest.py
   `-- wheelhouse (directory)
           |-- Appium_Python_Client-0.20-cp27-none-any.whl
           |-- py-1.4.31-py2.py3-none-any.whl
           |-- pytest-2.9.0-py2.py3-none-any.whl
           |-- selenium-2.52.0-cp27-none-any.whl
           `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_DEPENDENCY_WHEEL_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file roda ketergantungan di pohon direktori ruang kemudi. Harap unzip paket pengujian Anda dan kemudian buka direktori ruang kemudi, verifikasi bahwa setidaknya satu file roda ada di direktori, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Python valid, Anda akan menemukan setidaknya satu*.whl*file dependen seperti file yang disorot di dalam*ruang kemudi*direktori.

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `-- test_unittest.py
   `-- wheelhouse (directory)
           |-- Appium_Python_Client-0.20-cp27-none-any.whl
           |-- py-1.4.31-py2.py3-none-any.whl
           |-- pytest-2.9.0-py2.py3-none-any.whl
           |-- selenium-2.52.0-cp27-none-any.whl
           `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_INVALID_PLATFORM

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan setidaknya satu file roda menentukan platform yang tidak kami dukung. Harap unzip paket pengujian Anda dan kemudian buka direktori ruang kemudi, verifikasi bahwa nama file roda diakhiri dengan -any.whl atau -linux_x86_64.whl, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Python valid, Anda akan menemukan setidaknya satu*.whl*file dependen seperti file yang disorot di dalam*ruang kemudi*direktori. Nama file mungkin berbeda, tetapi harus diakhiri dengan*-apa saja.whl*atau*-linux_x86_64.whl*, yang menentukan platform. Platform lain sepertiwindowstidak didukung.

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `-- test_unittest.py
   `-- wheelhouse (directory)
           |-- Appium_Python_Client-0.20-cp27-none-any.whl
           |-- py-1.4.31-py2.py3-none-any.whl
           |-- pytest-2.9.0-py2.py3-none-any.whl
           |-- selenium-2.52.0-cp27-none-any.whl
           `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_TEST_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori tes di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori tes ada di dalam paket, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Python valid, Anda akan menemukan*tes*direktori di dalam direktori kerja.

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `-- test_unittest.py
   `-- wheelhouse (directory)
           |-- Appium_Python_Client-0.20-cp27-none-any.whl
           |-- py-1.4.31-py2.py3-none-any.whl
           |-- pytest-2.9.0-py2.py3-none-any.whl
           |-- selenium-2.52.0-cp27-none-any.whl
           `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_INVALID_TEST_FILE_NAME

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file pengujian yang valid di pohon direktori tes. Harap unzip paket pengujian Anda dan kemudian buka direktori tes, verifikasi bahwa setidaknya satu nama file dimulai atau diakhiri dengan kata kunci “test”, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Python valid, Anda akan menemukan*tes*direktori di dalam direktori kerja. Nama file mungkin berbeda, tetapi harus dimulai dengan*tes_*atau diakhiri dengan*_test.py*.

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `-- test_unittest.py
   `-- wheelhouse (directory)
           |-- Appium_Python_Client-0.20-cp27-none-any.whl
           |-- py-1.4.31-py2.py3-none-any.whl
           |-- pytest-2.9.0-py2.py3-none-any.whl
           |-- selenium-2.52.0-cp27-none-any.whl
           `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_REQUIREMENTS_TXT_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file requirements.txt di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa file requirements.txt ada di dalam paket, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Python valid, Anda akan menemukan*requirements.txt*berkas di dalam direktori kerja.

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `-- test_unittest.py
   `-- wheelhouse (directory)
           |-- Appium_Python_Client-0.20-cp27-none-any.whl
           |-- py-1.4.31-py2.py3-none-any.whl
           |-- pytest-2.9.0-py2.py3-none-any.whl
           |-- selenium-2.52.0-cp27-none-any.whl
           `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_INVALID_PYTEST_VERSION

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan versi pytest lebih rendah dari versi minimum 2.8.0 yang kami dukung. Harap ubah versi pytest di dalam file requirements.txt, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*requirements.txt*berkas di dalam direktori kerja.

   ```
   .
   |-- requirements.txt
   |-- test_bundle.zip
   |-- tests (directory)
   |       `--test_unittest.py
   `-- wheelhouse (directory)
          |-- Appium_Python_Client-0.20-cp27-none-any.whl
          |-- py-1.4.31-py2.py3-none-any.whl
          |-- pytest-2.9.0-py2.py3-none-any.whl
          |-- selenium-2.52.0-cp27-none-any.whl
          `-- wheel-0.26.0-py2.py3-none-any.whl
   ```

3. Untuk mendapatkan versi pytest, Anda dapat menjalankan perintah berikut:

   ```
   $ grep "pytest" requirements.txt
   ```

Anda harus menemukan output seperti berikut:

```
pytest==2.9.0
```

Ini menunjukkan versi pytest, yang dalam contoh ini adalah 2.9.0. Jika paket Appium Python valid, versi pytest harus lebih besar dari atau sama dengan 2.8.0.

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_INSTALL_DEPENDENCY_WHEELS_FAIL

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami gagal memasang roda ketergantungan. Harap unzip paket pengujian Anda dan kemudian buka file requirements.txt dan direktori ruang kemudi, verifikasi bahwa roda ketergantungan yang ditentukan dalam file requirements.txt sama persis dengan roda ketergantungan di dalam direktori ruang kemudi, dan coba lagi.

Kami sangat menyarankan agar Anda mengaturVirtualenv Pythonuntuk tes pengemasan. Berikut adalah contoh aliran menciptakan lingkungan virtual menggunakan Python virtualenv dan kemudian mengaktifkannya:

```
$ virtualenv workspace
$ cd workspace
$ source bin/activate
```

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Untuk menguji menginstal file roda, Anda dapat menjalankan perintah berikut:

```
$ pip install --use-wheel --no-index --find-links=./wheelhouse --requirement=./
requirements.txt
```

Paket Appium Python yang valid harus menghasilkan output seperti berikut:

```
Ignoring indexes: https://pypi.python.org/simple
Collecting Appium-Python-Client==0.20 (from -r ./requirements.txt (line 1))
Collecting py==1.4.31 (from -r ./requirements.txt (line 2))
Collecting pytest==2.9.0 (from -r ./requirements.txt (line 3))
Collecting selenium==2.52.0 (from -r ./requirements.txt (line 4))
Collecting wheel==0.26.0 (from -r ./requirements.txt (line 5))
Installing collected packages: selenium, Appium-Python-Client, py, pytest, wheel
  Found existing installation: wheel 0.29.0
    Uninstalling wheel-0.29.0:
      Successfully uninstalled wheel-0.29.0
Successfully installed Appium-Python-Client-0.20 py-1.4.31 pytest-2.9.0
 selenium-2.52.0 wheel-0.26.0
```

3. Untuk menonaktifkan lingkungan virtual, Anda dapat menjalankan perintah berikut:

```
$ deactivate
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_PYTEST_COLLECT_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami gagal mengumpulkan tes di direktori tes. Harap unzip paket pengujian Anda, sehingga paket pengujian valid dengan menjalankan perintah`py.test --collect-only <path to your tests directory>`, dan coba lagi setelah perintah tidak mencetak kesalahan apa pun.

Kami sangat menyarankan agar Anda mengaturVirtualenv Pythonuntuk tes pengemasan. Berikut adalah contoh aliran menciptakan lingkungan virtual menggunakan Python virtualenv dan kemudian mengaktifkannya:

```
$ virtualenv workspace
$ cd workspace
$ source bin/activate
```

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Untuk menginstal file roda, Anda dapat menjalankan perintah berikut:

   ```
   $ pip install --use-wheel --no-index --find-links=./wheelhouse --requirement=./
   requirements.txt
   ```

3. Untuk mengumpulkan tes, Anda dapat menjalankan perintah berikut:

   ```
   $ py.test --collect-only tests
   ```

   Paket Appium Python yang valid harus menghasilkan output seperti berikut:

   ```
   ==================== test session starts ====================
   platform darwin -- Python 2.7.11, pytest-2.9.0, py-1.4.31, pluggy-0.3.1
   rootdir: /Users/zhena/Desktop/Ios/tests, inifile:
   collected 1 items
   <Module 'test_unittest.py'>
     <UnitTestCase 'DeviceFarmAppiumWebTests'>
       <TestCaseFunction 'test_devicefarm'>

   ==================== no tests ran in 0.11 seconds ====================
   ```

4. Untuk menonaktifkan lingkungan virtual, Anda dapat menjalankan perintah berikut:

   ```
   $ deactivate
   ```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_PYTHON_TEST_PACKAGE_DEPENDENCY_WHEELS_INSUFFICIEN

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan dependensi roda yang cukup di direktori ruang kemudi. Silakan unzip paket pengujian Anda, lalu buka direktori ruang kemudi. Verifikasi bahwa Anda memiliki semua dependensi roda yang ditentukan dalam file requirements.txt.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Periksa panjang*requirements.txt*berkas serta jumlah*.whl*file dependen di direktori ruang kemudi:

   ```
   $ cat requirements.txt | egrep "." |wc -l
       12
   $ ls wheelhouse/ | egrep ".+\.whl" | wc -l
       11
   ```

   Jika jumlah*.whl*file dependen kurang dari jumlah baris yang tidak kosong di*requirements.txt*file, maka Anda perlu memastikan yang berikut:

   - Ada*.whl*file dependen yang sesuai dengan setiap baris di*requirements.txt*berkas.
   - Tidak ada garis lain di*requirements.txt*file yang berisi informasi selain nama paket dependensi.
   - Tidak ada nama ketergantungan yang diduplikasi dalam beberapa baris di*requirements.txt*berkas sedemikian rupa sehingga dua baris dalam file mungkin sesuai dengan satu*.whl*file tergantung.

AWS Device Farm tidak mendukung baris di*requirements.txt*file yang tidak secara langsung sesuai dengan paket dependensi, seperti baris yang menentukan opsi global untuk`pip install`perintah. LihatFormat file persyaratanuntuk daftar opsi global.

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

Memecahkan masalah pengujian aplikasi web Appium Python di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian aplikasi Web Appium Python dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

APPIUM_WEB_PYTHON_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP uji Appium Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket Appium Python yang valid harus menghasilkan output seperti berikut:

   ```
   .
   ```

```
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `-- test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_DEPENDENCY_WHEEL_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file roda ketergantungan di pohon direktori ruang kemudi. Harap unzip paket pengujian Anda dan kemudian buka direktori ruang kemudi, verifikasi bahwa setidaknya satu file roda ada di direktori, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket Appium Python valid, Anda akan menemukan setidaknya satu*.whl*file dependen seperti file yang disorot di dalam*ruang kemudi*direktori.

```
.
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `-- test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_INVALID_PLATFORM

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan setidaknya satu file roda menentukan platform yang tidak kami dukung. Harap unzip paket pengujian Anda dan kemudian buka direktori ruang kemudi, verifikasi bahwa nama file roda diakhiri dengan -any.whl atau -linux_x86_64.whl, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

Jika paket Appium Python valid, Anda akan menemukan setidaknya satu*.whl*file dependen seperti file yang disorot di dalam*ruang kemudi*direktori. Nama file mungkin berbeda, tetapi harus diakhiri dengan*-apa saja.whl*atau*-linux_x86_64.whl*, yang menentukan platform. Platform lain sepertiwindowstidak didukung.

```
.
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `-- test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_TEST_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori tes di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori tes ada di dalam paket, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip test_bundle.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Jika paket Appium Python valid, Anda akan menemukan*tes*direktori di dalam direktori kerja.

```
.
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `-- test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_INVALID_TEST_FILE_NAME

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file pengujian yang valid di pohon direktori tes. Harap unzip paket pengujian Anda dan kemudian buka direktori tes, verifikasi bahwa setidaknya satu nama file dimulai atau diakhiri dengan kata kunci “test”, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Jika paket Appium Python valid, Anda akan menemukan*tes*direktori di dalam direktori kerja. Nama file mungkin berbeda, tetapi harus dimulai dengan*ujian_*atau diakhiri dengan*_test.py*.

```
.
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `-- test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_REQUIREMENTS_TXT_FILE_MISS

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file requirements.txt di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa file requirements.txt ada di dalam paket, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip test_bundle.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Jika paket Appium Python valid, Anda akan menemukan*requirements.txt*berkas di dalam direktori kerja.

```
.
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `-- test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_INVALID_PYTEST_VERSION

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan versi pytest lebih rendah dari versi minimum 2.8.0 yang kami dukung. Harap ubah versi pytest di dalam file requirements.txt, dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip test_bundle.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*requirement.txt*berkas di dalam direktori kerja.

```
.
|-- requirements.txt
|-- test_bundle.zip
|-- tests (directory)
|       `--test_unittest.py
`-- wheelhouse (directory)
        |-- Appium_Python_Client-0.20-cp27-none-any.whl
        |-- py-1.4.31-py2.py3-none-any.whl
        |-- pytest-2.9.0-py2.py3-none-any.whl
        |-- selenium-2.52.0-cp27-none-any.whl
        `-- wheel-0.26.0-py2.py3-none-any.whl
```

3. Untuk mendapatkan versi pytest, Anda dapat menjalankan perintah berikut:

```
$ grep "pytest" requirements.txt
```

Anda harus menemukan output seperti berikut:

```
pytest==2.9.0
```

Ini menunjukkan versi pytest, yang dalam contoh ini adalah 2.9.0. Jika paket Appium Python valid, versi pytest harus lebih besar dari atau sama dengan 2.8.0.

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_INSTALL_DEPENDENCY_WHEELS

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami gagal memasang roda ketergantungan. Harap unzip paket pengujian Anda dan kemudian buka file requirements.txt dan direktori ruang kemudi, verifikasi bahwa roda ketergantungan yang ditentukan dalam file requirements.txt sama persis dengan roda ketergantungan di dalam direktori ruang kemudi, dan coba lagi.

Kami sangat menyarankan agar Anda mengaturVirtualenv Pythonuntuk tes pengemasan. Berikut adalah contoh aliran menciptakan lingkungan virtual menggunakan Python virtualenv dan kemudian mengaktifkannya:

```
$ virtualenv workspace
$ cd workspace
$ source bin/activate
```

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip test_bundle.zip
   ```

2. Untuk menguji menginstal file roda, Anda dapat menjalankan perintah berikut:

   ```
   $ pip install --use-wheel --no-index --find-links=./wheelhouse --requirement=./
   requirements.txt
   ```

   Paket Appium Python yang valid harus menghasilkan output seperti berikut:

   ```
   Ignoring indexes: https://pypi.python.org/simple
   Collecting Appium-Python-Client==0.20 (from -r ./requirements.txt (line 1))
   Collecting py==1.4.31 (from -r ./requirements.txt (line 2))
   Collecting pytest==2.9.0 (from -r ./requirements.txt (line 3))
   Collecting selenium==2.52.0 (from -r ./requirements.txt (line 4))
   Collecting wheel==0.26.0 (from -r ./requirements.txt (line 5))
   Installing collected packages: selenium, Appium-Python-Client, py, pytest, wheel
     Found existing installation: wheel 0.29.0
       Uninstalling wheel-0.29.0:
         Successfully uninstalled wheel-0.29.0
   ```

```
Successfully installed Appium-Python-Client-0.20 py-1.4.31 pytest-2.9.0
 selenium-2.52.0 wheel-0.26.0
```

3. Untuk menonaktifkan lingkungan virtual, Anda dapat menjalankan perintah berikut:

```
$ deactivate
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

APPIUM_WEB_PYTHON_TEST_PACKAGE_PYTEST_COLLECT_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami gagal mengumpulkan tes di direktori tes. Harap unzip paket pengujian Anda, sehingga paket pengujian valid dengan menjalankan perintah “py.test --collect-only <path to your tests directory>“, dan coba lagi setelah perintah tidak mencetak kesalahan apa pun.

Kami sangat menyarankan agar Anda mengaturVirtualenv Pythonuntuk tes pengemasan. Berikut adalah contoh aliran menciptakan lingkungan virtual menggunakan Python virtualenv dan kemudian mengaktifkannya:

```
$ virtualenv workspace
$ cd workspace
$ source bin/activate
```

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahtest_bundle.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip test_bundle.zip
```

2. Untuk menginstal file roda, Anda dapat menjalankan perintah berikut:

```
$ pip install --use-wheel --no-index --find-links=./wheelhouse --requirement=./
requirements.txt
```

3. Untuk mengumpulkan tes, Anda dapat menjalankan perintah berikut:

```
$ py.test --collect-only tests
```

Paket Appium Python yang valid harus menghasilkan output seperti berikut:

```
=================== test session starts ====================
platform darwin -- Python 2.7.11, pytest-2.9.0, py-1.4.31, pluggy-0.3.1
rootdir: /Users/zhena/Desktop/Ios/tests, inifile:
collected 1 items
<Module 'test_unittest.py'>
  <UnitTestCase 'DeviceFarmAppiumWebTests'>
    <TestCaseFunction 'test_devicefarm'>

=================== no tests ran in 0.11 seconds ====================
```

4. Untuk menonaktifkan lingkungan virtual, Anda dapat menjalankan perintah berikut:

```
$ deactivate
```

Untuk informasi selengkapnya, lihat Bekerja dengan Appium dan AWS Device Farm.

Memecahkan masalah pengujian instrumentasi di AWS Device Farm

Topik berikut mencantumkan pesan galat yang terjadi selama pengunggahan pengujian Instrumentasi dan merekomendasikan solusi untuk mengatasi setiap kesalahan.

INSTRUMENTATION_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file APK pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat membuka zip paket pengujian tanpa kesalahan. Dalam contoh berikut, nama paket adalahapp-debug-androidTest-unaligned.apk.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip app-debug-androidTest-unaligned.apk
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Paket uji Instrumentasi yang valid akan menghasilkan output seperti berikut:

```
.
|-- AndroidManifest.xml
|--  classes.dex
|-- resources.arsc
|-- LICENSE-junit.txt
|-- junit (directory)
`-- META-INF (directory)
```

Untuk informasi selengkapnya, lihat Bekerja dengan instrumentasi untuk Android dan AWS Device Farm.

INSTRUMENTATION_TEST_PACKAGE_AAPT_DEBUG_BADGING_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat mengekstrak informasi tentang paket pengujian Anda. Harap verifikasi bahwa paket pengujian valid dengan menjalankan perintah “aapt debug badging <path to your test package>“, dan coba lagi setelah perintah tidak mencetak kesalahan apa pun.

Selama proses validasi unggahan, Device Farm mem-parsing informasi dari output`aapt debug badging <path to your package>`perintah.

Pastikan Anda dapat menjalankan perintah ini pada paket pengujian Instrumentasi Anda dengan sukses.

Dalam contoh berikut, nama paket adalahapp-debug-androidTest-unaligned.apk.

- Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ aapt debug badging app-debug-androidTest-unaligned.apk
```

Paket uji Instrumentasi yang valid akan menghasilkan output seperti berikut:

```
package: name='com.amazon.aws.adf.android.referenceapp.test' versionCode=''
 versionName='' platformBuildVersionName='5.1.1-1819727'
sdkVersion:'9'
targetSdkVersion:'22'
application-label:'Test-api'
application: label='Test-api' icon=''
application-debuggable
uses-library:'android.test.runner'
feature-group: label=''
uses-feature: name='android.hardware.touchscreen'
uses-implied-feature: name='android.hardware.touchscreen' reason='default feature
 for all apps'
supports-screens: 'small' 'normal' 'large' 'xlarge'
supports-any-density: 'true'
locales: '--_--'
densities: '160'
```

Untuk informasi selengkapnya, lihat Bekerja dengan instrumentasi untuk Android dan AWS Device Farm.

INSTRUMENTATION_TEST_PACKAGE_INSTRUMENTATION_RUNNER_VALU

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai runner instrumentasi diAndroidManifest.xml.xl. Harap verifikasi bahwa paket pengujian valid dengan menjalankan perintah “aapt dump xmltree

<path to your test package>AndroidManifest.xml"dan coba lagi setelah menemukan nilai instrumentasi runner di belakang kata kunci “instrumentasi.”

Selama proses validasi upload, Device Farm mem-parsing nilai instrumentasi runner dari pohon parse XMLuntuk file XMLyang terdapat di dalam paket. Anda dapat menggunakan perintah berikut:`aapt dump xmltree <path to your package> AndroidManifest.xml`.

Pastikan Anda dapat menjalankan perintah ini pada paket pengujian Instrumentasi Anda dan menemukan nilai instrumentasi dengan sukses.

Dalam contoh berikut, nama paket adalahapp-debug-androidTest-unaligned.apk.

- Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ aapt dump xmltree app-debug-androidTest-unaligned.apk AndroidManifest.xml | grep
 -A5 "instrumentation"
```

Paket uji Instrumentasi yang valid akan menghasilkan output seperti berikut:

```
E: instrumentation (line=9)
      A: android:label(0x01010001)="Tests for
 com.amazon.aws.adf.android.referenceapp" (Raw: "Tests for
 com.amazon.aws.adf.android.referenceapp")
      A:
 android:name(0x01010003)="android.support.test.runner.AndroidJUnitRunner" (Raw:
 "android.support.test.runner.AndroidJUnitRunner")
      A:
 android:targetPackage(0x01010021)="com.amazon.aws.adf.android.referenceapp" (Raw:
 "com.amazon.aws.adf.android.referenceapp")
      A: android:handleProfiling(0x01010022)=(type 0x12)0x0
      A: android:functionalTest(0x01010023)=(type 0x12)0x0
```

Untuk informasi selengkapnya, lihat Bekerja dengan instrumentasi untuk Android dan AWS Device Farm.

INSTRUMENTATION_TEST_PACKAGE_AAPT_DUMP_XMLTREE_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan yang validAndroidManifest.xml. dalam paket pengujian Anda. Harap verifikasi bahwa paket pengujian valid dengan menjalankan perintah “aapt dump xmltree <path to your test package>AndroidManifest.xml"dan coba lagi setelah perintah tidak mencetak kesalahan apa pun.

Selama proses validasi upload, Device Farm mem-parsing informasi dari pohon parse XMLuntuk file XMLyang terdapat dalam paket menggunakan perintah berikut:`aapt dump xmltree <path to your package> AndroidManifest.xml`.

Pastikan bahwa Anda dapat menjalankan perintah ini pada paket pengujian instrumentasi Anda berhasil.

Dalam contoh berikut, nama paket adalahapp-debug-androidTest-unaligned.apk.

- Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

  ```
  $ aapt dump xmltree app-debug-androidTest-unaligned.apk AndroidManifest.xml
  ```

  Paket uji Instrumentasi yang valid akan menghasilkan output seperti berikut:

  ```
  N: android=http://schemas.android.com/apk/res/android
    E: manifest (line=2)
      A: package="com.amazon.aws.adf.android.referenceapp.test" (Raw:
   "com.amazon.aws.adf.android.referenceapp.test")
      A: platformBuildVersionCode=(type 0x10)0x16 (Raw: "22")
      A: platformBuildVersionName="5.1.1-1819727" (Raw: "5.1.1-1819727")
      E: uses-sdk (line=5)
        A: android:minSdkVersion(0x0101020c)=(type 0x10)0x9
        A: android:targetSdkVersion(0x01010270)=(type 0x10)0x16
      E: instrumentation (line=9)
        A: android:label(0x01010001)="Tests for
   com.amazon.aws.adf.android.referenceapp" (Raw: "Tests for
   com.amazon.aws.adf.android.referenceapp")
        A:
   android:name(0x01010003)="android.support.test.runner.AndroidJUnitRunner" (Raw:
   "android.support.test.runner.AndroidJUnitRunner")
  ```

```
      A:
android:targetPackage(0x01010021)="com.amazon.aws.adf.android.referenceapp" (Raw:
"com.amazon.aws.adf.android.referenceapp")
      A: android:handleProfiling(0x01010022)=(type 0x12)0x0
      A: android:functionalTest(0x01010023)=(type 0x12)0x0
    E: application (line=16)
      A: android:label(0x01010001)=@0x7f020000
      A: android:debuggable(0x0101000f)=(type 0x12)0xffffffff
      E: uses-library (line=17)
        A: android:name(0x01010003)="android.test.runner" (Raw:
"android.test.runner")
```

Untuk informasi selengkapnya, lihat Bekerja dengan instrumentasi untuk Android dan AWS Device Farm.

INSTRUMENTASI_TEST_PACKAGE_TEST_PACKAGE_NAME_VALUE_MISSIN

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nama paket dalam paket pengujian Anda. Harap verifikasi bahwa paket pengujian valid dengan menjalankan perintah “aapt debug badging <path to your test package>“, dan coba lagi setelah menemukan nilai nama paket di belakang kata kunci “package: name.”

Selama proses validasi upload, Device Farm mem-parsing nilai nama paket dari output perintah berikut:`aapt debug badging <path to your package>`.

Pastikan bahwa Anda dapat menjalankan perintah ini pada paket pengujian Instrumentasi Anda dan menemukan nilai nama paket berhasil.

Dalam contoh berikut, nama paket adalahapp-debug-androidTest-unaligned.apk.

- Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

  ```
  $ aapt debug badging app-debug-androidTest-unaligned.apk | grep "package: name="
  ```

  Paket uji Instrumentasi yang valid akan menghasilkan output seperti berikut:

```
package: name='com.amazon.aws.adf.android.referenceapp.test' versionCode=''
 versionName='' platformBuildVersionName='5.1.1-1819727'
```

Untuk informasi selengkapnya, lihat Bekerja dengan instrumentasi untuk Android dan AWS Device Farm.

Memecahkan masalah pengujian aplikasi iOS di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian aplikasi iOS dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

> **Note**
>
> Petunjuk di bawah ini didasarkan pada Linux x86_64 dan Mac.

IOS_APP_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka aplikasi Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat unzip paket aplikasi tanpa kesalahan. Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

```
.
`-- Payload (directory)
        `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                        |-- Info.plist
                        `-- (any other files)
```

Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_PAYLOAD_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori Payload di dalam aplikasi Anda. Silakan unzip aplikasi Anda, verifikasi bahwa direktori Payload ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket aplikasi iOS valid, Anda akan menemukan*Muatan*direktori di dalam direktori kerja.

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                           |-- Info.plist
                           `-- (any other files)
   ```

Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_APP_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori.app di dalam direktori Payload. Silakan unzip aplikasi Anda dan kemudian buka direktori Payload, verifikasi bahwa direktori.app ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket aplikasi iOS valid, Anda akan menemukan*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita di dalam*Muatan*direktori.

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                       |-- Info.plist
                       `-- (any other files)
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_PLIST_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file Info.plist di dalam direktori.app. Silakan unzip aplikasi Anda dan kemudian buka direktori.app, verifikasi bahwa file Info.plist ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket aplikasi iOS valid, Anda akan menemukan*Info.plist*berkas di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita.

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                       |-- Info.plist
                       `-- (any other files)
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_CPU_ARCHITECTURE_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai arsitektur CPU di file Info.plist. Silakan unzip aplikasi Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “UIRequiredDeviceCapabilities“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                       |-- Info.plist
                       `-- (any other files)
   ```

3. Untuk menemukan nilai arsitektur CPU, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/AWSDeviceFarmiOSReferenceApp-cal.app/
   Info.plist')
   print info_plist['UIRequiredDeviceCapabilities']
   ```

Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

```
['armv7']
```

Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_PLATFORM_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai platform di file Info.plist. Silakan unzip aplikasi Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “CFBundleSupportedPlatforms“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                       |-- Info.plist
                       `-- (any other files)
   ```

3. Untuk menemukan nilai platform, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/AWSDeviceFarmiOSReferenceApp-cal.app/
   Info.plist')
   print info_plist['CFBundleSupportedPlatforms']
   ```

   Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

   ```
   ['iPhoneOS']
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_WRONG_PLATFORM_DEVICE_VALUE

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan nilai perangkat platform salah dalam file Info.plist. Silakan unzip aplikasi Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa nilai kunci “CFBundleSupportedPlatforms“tidak mengandung kata kunci “simulator”, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita:

```
.
`-- Payload (directory)
        `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                        |-- Info.plist
                        `-- (any other files)
```

3. Untuk menemukan nilai platform, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/AWSDeviceFarmiOSReferenceApp-cal.app/
   Info.plist')
   print info_plist['CFBundleSupportedPlatforms']
   ```

   Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

   ```
   ['iPhoneOS']
   ```

   Jika aplikasi iOS valid, nilainya tidak boleh mengandung kata kunci`simulator`.

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_FORM_FACTOR_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai faktor bentuk dalam file Info.plist. Silakan unzip aplikasi Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “UIDeviceFamily“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                       |-- Info.plist
                       `-- (any other files)
   ```

3. Untuk menemukan nilai faktor bentuk, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/AWSDeviceFarmiOSReferenceApp-cal.app/
   Info.plist')
   print info_plist['UIDeviceFamily']
   ```

Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

```
[1, 2]
```

Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_PACKAGE_NAME_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai nama paket di file Info.plist. Silakan unzip aplikasi Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “CFBundleIdentifier“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                       |-- Info.plist
                       `-- (any other files)
   ```

3. Untuk menemukan nilai nama paket, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/AWSDeviceFarmiOSReferenceApp-cal.app/
   Info.plist')
   print info_plist['CFBundleIdentifier']
   ```

   Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

   ```
   Amazon.AWSDeviceFarmiOSReferenceApp
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

IOS_APP_EXECUTABLE_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai yang dapat dieksekusi di file Info.plist. Silakan unzip aplikasi Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “CFBundleExecutable“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahAWSDeviceFarmiOSReferenceApp.ipa.

1. Salin paket aplikasi Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip AWSDeviceFarmiOSReferenceApp.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*AWSDeviceFarmiOSReferenceApp.aplikasi*dalam contoh kita:

```
.
`-- Payload (directory)
        `-- AWSDeviceFarmiOSReferenceApp.app (directory)
                        |-- Info.plist
                        `-- (any other files)
```

3. Untuk menemukan nilai yang dapat dieksekusi, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/AWSDeviceFarmiOSReferenceApp-cal.app/
   Info.plist')
   print info_plist['CFBundleExecutable']
   ```

   Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

   ```
   AWSDeviceFarmiOSReferenceApp
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan pengujian iOS di AWS Device Farm.

Memecahkan masalah pengujian XCTest di AWS Device Farm

Topik berikut mencantumkan pesan kesalahan yang terjadi selama pengunggahan pengujian XCTest dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

> **Note**
>
> Petunjuk di bawah ini mengasumsikan Anda menggunakan macOS.

XCTEST_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file ZIP pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat unzip paket aplikasi tanpa kesalahan. Dalam contoh berikut, nama paket adalahswiftExampleTests.xctest-1.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swiftExampleTests.xctest-1.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket XCTest yang valid harus menghasilkan output seperti berikut:

   ```
   .
   `-- swiftExampleTests.xctest (directory)
               |-- Info.plist
               `-- (any other files)
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan XCTest untuk iOS dan AWS Device Farm.

XCTEST_TEST_PACKAGE_XCTEST_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori.xctest di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori.xctest ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahswiftExampleTests.xctest-1.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swiftExampleTests.xctest-1.zip
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest valid, Anda akan menemukan direktori dengan nama yang mirip dengan*swiftExampleTests.xctest*di dalam direktori kerja. Nama harus diakhiri dengan*.xctest*.

   ```
   .
   `-- swiftExampleTests.xctest (directory)
                |-- Info.plist
                `-- (any other files)
   ```

   Untuk informasi selengkapnya, lihat Bekerja dengan XCTest untuk iOS dan AWS Device Farm.

XCTEST_TEST_PACKAGE_PLIST_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file Info.plist di dalam direktori.xctest. Silakan unzip paket pengujian Anda dan kemudian buka direktori.xctest, verifikasi bahwa file Info.plist ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahswiftExampleTests.xctest-1.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip swiftExampleTests.xctest-1.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Jika paket XCTest valid, Anda akan menemukan*Info.plist*berkas di dalam*.xctest*direktori. Dalam contoh kita di bawah ini, direktori disebut*swiftExampleTests.xctest*.

```
.
`-- swiftExampleTests.xctest (directory)
              |-- Info.plist
              `-- (any other files)
```

Untuk informasi selengkapnya, lihat Bekerja dengan XCTest untuk iOS dan AWS Device Farm.

XCTEST_TEST_PACKAGE_PACKAGE_NAME_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai nama paket di file Info.plist. Silakan unzip paket pengujian Anda dan kemudian buka file Info.plist, verifikasi bahwa kunci “CFBundleIdentifier“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahswiftExampleTests.xctest-1.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip swiftExampleTests.xctest-1.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*Info.plist*file di dalam*.xctest*direktori seperti*swiftExampleTests.xctest*dalam contoh kita:

```
.
`-- swiftExampleTests.xctest (directory)
              |-- Info.plist
              `-- (any other files)
```

3. Untuk menemukan nilai nama paket, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

```
$ pip install biplist
```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('swiftExampleTests.xctest/Info.plist')
print info_plist['CFBundleIdentifier']
```

Paket aplikasi XCTest yang valid harus menghasilkan output seperti berikut:

```
com.amazon.kanapka.swiftExampleTests
```

Untuk informasi selengkapnya, lihat Bekerja dengan XCTest untuk iOS dan AWS Device Farm.

XCTEST_TEST_PACKAGE_EXECUTABLE_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai yang dapat dieksekusi di file Info.plist. Silakan unzip paket pengujian Anda dan kemudian buka file Info.plist, verifikasi bahwa kunci "CFBundleExecutable"ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahswiftExampleTests.xctest-1.zip.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip swiftExampleTests.xctest-1.zip
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

   Anda harus menemukan*Info.plist*file di dalam*.xctest*direktori seperti*swiftExampleTests.xctest*dalam contoh kita:

```
.
`-- swiftExampleTests.xctest (directory)
              |-- Info.plist
              `-- (any other files)
```

3. Untuk menemukan nilai nama paket, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

```
$ pip install biplist
```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('swiftExampleTests.xctest/Info.plist')
print info_plist['CFBundleExecutable']
```

   Paket aplikasi XCTest yang valid harus menghasilkan output seperti berikut:

```
swiftExampleTests
```

Untuk informasi selengkapnya, lihat Bekerja dengan XCTest untuk iOS dan AWS Device Farm.

Memecahkan masalah pengujian UI XCTest di AWS Device Farm

Topik berikut mencantumkan pesan galat yang terjadi selama pengunggahan pengujian UI XCTest dan merekomendasikan solusi untuk menyelesaikan setiap kesalahan.

> **Note**
>
> Petunjuk di bawah ini didasarkan pada Linux x86_64 dan Mac.

XCTEST_UI_TEST_PACKAGE_UNZIP_FAILED

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat membuka file IPA pengujian Anda. Harap verifikasi bahwa file tersebut valid dan coba lagi.

Pastikan Anda dapat unzip paket aplikasi tanpa kesalahan. Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Paket aplikasi iOS yang valid harus menghasilkan output seperti berikut:

```
.
`-- Payload (directory)
        `-- swift-sampleUITests-Runner.app (directory)
                        |-- Info.plist
                        |-- Plugins (directory)
                        |       `swift-sampleUITests.xctest (directory)
                        |                         |-- Info.plist
                        |                         `-- (any other files)
                        `-- (any other files)
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_PAYLOAD_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori Payload di dalam paket pengujian Anda. Harap unzip paket pengujian Anda, verifikasi bahwa direktori Payload ada di dalam paket, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest UI valid, Anda akan menemukan*Muatan*direktori di dalam direktori kerja.

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                           |-- Info.plist
   ```

```
                    |-- Plugins (directory)
                    |       `swift-sampleUITests.xctest (directory)
                    |                       |-- Info.plist
                    |                       `-- (any other files)
                    `-- (any other files)
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_APP_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori.app di dalam direktori Payload. Harap unzip paket pengujian Anda dan kemudian buka direktori Payload, verifikasi bahwa direktori.app ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest UI valid, Anda akan menemukan*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita di dalam*Muatan*direktori.

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
   ```

```
                    |                       |-- Info.plist
                    |                       `-- (any other files)
                    `-- (any other files)
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_PLUGINS_DIR_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori Plugins di dalam direktori.app. Silakan unzip paket pengujian Anda dan kemudian buka direktori.app, verifikasi bahwa direktori Plugins ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest UI valid, Anda akan menemukan*Plugin*direktori di dalam*.aplikasi*direktori. Dalam contoh kita, direktori disebut*Swift-sampleuitests-runner.app*.

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                       |-- Info.plist
   ```

```
                    |                       `-- (any other files)
                    `-- (any other files)
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_XCTEST_DIR_MISSING_IN_PLUGINS_DIR

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan direktori.xctest di dalam direktori plugin. Silakan unzip paket pengujian Anda dan kemudian buka direktori plugin, verifikasi bahwa direktori.xctest ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest UI valid, Anda akan menemukan*.xctest*direktori di dalam*Plugin*direktori. Dalam contoh kita, direktori disebut*Swift-sampleuitests.xctest*.

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                       |-- Info.plist
                       |                       `-- (any other files)
                       `-- (any other files)
   ```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_PLIST_FILE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file Info.plist di dalam direktori.app. Silakan unzip paket pengujian Anda dan kemudian buka direktori.app, verifikasi bahwa file Info.plist ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest UI valid, Anda akan menemukan*Info.plist*berkas di dalam*.aplikasi*direktori. Dalam contoh kita di bawah ini, direktori disebut*Swift-sampleuitests-runner.app*.

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                         |-- Info.plist
                       |                         `-- (any other files)
                       `-- (any other files)
   ```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_PLIST_FILE_MISSING_IN_XCTEST_DIR

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan file Info.plist di dalam direktori.xctest. Silakan unzip paket pengujian Anda dan kemudian buka direktori.xctest, verifikasi bahwa file Info.plist ada di dalam direktori, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Jika paket XCTest UI valid, Anda akan menemukan*Info.plist*berkas di dalam*.xctest*direktori. Dalam contoh kita di bawah ini, direktori disebut*Swift-sampleuitests.xctest*.

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                         |-- Info.plist
                       |                         `-- (any other files)
                       `-- (any other files)
   ```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_CPU_ARCHITECTURE_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak bisa nilai arsitektur CPU dalam file Info.plist. Harap unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “UIRequiredDeviceCapabilities“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                       |-- Info.plist
                       |                       `-- (any other files)
                       `-- (any other files)
   ```

3. Untuk menemukan nilai arsitektur CPU, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

```
$ pip install biplist
```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/
Info.plist')
print info_plist['UIRequiredDeviceCapabilities']
```

Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

```
['armv7']
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_PLATFORM_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai platform di Info.plist. Harap unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “CFBundleSupportedPlatforms“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip swift-sample-UI.ipa
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

```
.
`-- Payload (directory)
        `-- swift-sampleUITests-Runner.app (directory)
                        |-- Info.plist
                        |-- Plugins (directory)
                        |       `swift-sampleUITests.xctest (directory)
                        |                         |-- Info.plist
                        |                         `-- (any other files)
                        `-- (any other files)
```

3. Untuk menemukan nilai platform, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Info.plist')
   print info_plist['CFBundleSupportedPlatforms']
   ```

   Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

   ```
   ['iPhoneOS']
   ```

   Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_WRONG_PLATFORM_DEVICE_VALUE

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami menemukan nilai perangkat platform salah dalam file Info.plist. Harap unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa nilai kunci “CFBundleSupportedPlatforms“tidak mengandung kata kunci “simulator”, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*berkas di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                         |-- Info.plist
                       |                         `-- (any other files)
                       `-- (any other files)
   ```

3. Untuk menemukan nilai platform, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Info.plist')
print info_plist['CFBundleSupportedPlatforms']
```

Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

```
['iPhoneOS']
```

Jika paket XCTest UI valid, nilainya tidak boleh berisi kata kuncisimulator.

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_FORM_FACTOR_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak bisa mendapatkan nilai faktor bentuk di Info.plist. Harap unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “UIDeviceFamily“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*berkas di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

   ```
   .
   ```

```
`-- Payload (directory)
        `-- swift-sampleUITests-Runner.app (directory)
                        |-- Info.plist
                        |-- Plugins (directory)
                        |       `swift-sampleUITests.xctest (directory)
                        |                         |-- Info.plist
                        |                         `-- (any other files)
                        `-- (any other files)
```

3. Untuk menemukan nilai faktor bentuk, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Info.plist')
   print info_plist['UIDeviceFamily']
   ```

   Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

   ```
   [1, 2]
   ```

   Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_PACKAGE_NAME_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai nama paket di file Info.plist. Harap unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “CFBundleIdentifier“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip swift-sample-UI.ipa
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*Info.plist*berkas di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

```
.
`-- Payload (directory)
        `-- swift-sampleUITests-Runner.app (directory)
                      |-- Info.plist
                      |-- Plugins (directory)
                      |       `swift-sampleUITests.xctest (directory)
                      |                       |-- Info.plist
                      |                       `-- (any other files)
                      `-- (any other files)
```

3. Untuk menemukan nilai nama paket, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

```
$ pip install biplist
```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Info.plist')
print info_plist['CFBundleIdentifier']
```

Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

```
com.apple.test.swift-sampleUITests-Runner
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_EXECUTABLE_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai yang dapat dieksekusi di file Info.plist. Harap unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.app, verifikasi bahwa kunci “CFBundleExecutable“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*berkas di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                       |-- Info.plist
                       |                       `-- (any other files)
                       `-- (any other files)
   ```

3. Untuk menemukan nilai yang dapat dieksekusi, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

```
$ pip install biplist
```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Info.plist')
print info_plist['CFBundleExecutable']
```

Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

```
XCTRunner
```

Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_TEST_PACKAGE_NAME_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai nama paket di file Info.plist di dalam direktori.xctest. Silakan unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.xctest, verifikasi bahwa kunci “CFBundleIdentifier“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

```
$ unzip swift-sample-UI.ipa
```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

```
$ tree .
```

Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

```
.
`-- Payload (directory)
        `-- swift-sampleUITests-Runner.app (directory)
                        |-- Info.plist
                        |-- Plugins (directory)
                        |       `swift-sampleUITests.xctest (directory)
                        |                         |-- Info.plist
                        |                         `-- (any other files)
                        `-- (any other files)
```

3. Untuk menemukan nilai nama paket, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

   ```
   import biplist
   info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Plugins/
   swift-sampleUITests.xctest/Info.plist')
   print info_plist['CFBundleIdentifier']
   ```

   Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

   ```
   com.amazon.swift-sampleUITests
   ```

   Untuk informasi selengkapnya, lihat XCTest UI.

XCTEST_UI_TEST_PACKAGE_TEST_EXECUTABLE_VALUE_MISSING

Jika Anda melihat pesan berikut, ikuti langkah-langkah berikut untuk memperbaiki masalah.

> **Warning**
>
> Kami tidak dapat menemukan nilai yang dapat dieksekusi dalam file Info.plist di dalam direktori.xctest. Silakan unzip paket pengujian Anda dan kemudian buka file Info.plist di dalam direktori.xctest, verifikasi bahwa kunci “CFBundleExecutable“ditentukan, dan coba lagi.

Dalam contoh berikut, nama paket adalahSwift-sample-ui.ipa.

1. Salin paket pengujian Anda ke direktori kerja Anda, lalu jalankan perintah berikut:

   ```
   $ unzip swift-sample-UI.ipa
   ```

2. Setelah Anda berhasil unzip paket, Anda dapat menemukan struktur pohon direktori kerja dengan menjalankan perintah berikut:

   ```
   $ tree .
   ```

   Anda harus menemukan*Info.plist*file di dalam*.aplikasi*direktori seperti*Swift-sampleuitests-runner.app*dalam contoh kita:

   ```
   .
   `-- Payload (directory)
           `-- swift-sampleUITests-Runner.app (directory)
                       |-- Info.plist
                       |-- Plugins (directory)
                       |       `swift-sampleUITests.xctest (directory)
                       |                         |-- Info.plist
                       |                         `-- (any other files)
                       `-- (any other files)
   ```

3. Untuk menemukan nilai yang dapat dieksekusi, Anda dapat membuka Info.plist menggunakan Xcode atau Python.

   Untuk Python, Anda dapat menginstal modul biplist dengan menjalankan perintah berikut:

   ```
   $ pip install biplist
   ```

4. Selanjutnya, buka Python dan jalankan perintah berikut:

```
import biplist
info_plist = biplist.readPlist('Payload/swift-sampleUITests-Runner.app/Plugins/
swift-sampleUITests.xctest/Info.plist')
print info_plist['CFBundleExecutable']
```

Paket XCTest UI yang valid harus menghasilkan output seperti berikut:

```
swift-sampleUITests
```

Untuk informasi selengkapnya, lihat XCTest UI.

Keamanan di AWS Device Farm

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda akan mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menggambarkan ini sebagai keamanan dari cloud dan keamanan di dalam cloud:

- Keamanan cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan-layanan AWS di dalam AWS Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga menguji dan memverifikasi efektivitas keamanan kami secara berkala sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlakuAWS Device Farm, lihatLayanan AWS dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud – Tanggung jawab Anda ditentukan menurut layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta hukum dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Device Farm. Topik berikut menunjukkan cara mengonfigurasi Device Farm untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan layanan AWS lain yang membantu Anda memantau dan mengamankan sumber daya Device Farm Anda.

Topik

- Praktik terbaik keamanan untuk Device Farm

Manajemen identitas dan akses di AWS Device Farm

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Device Farm.

Pengguna layanan — Jika Anda menggunakan layanan Device Farm untuk melakukan pekerjaan Anda, administrator Anda akan memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Device Farm untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Device Farm, lihatMemecahkan masalah identitas dan akses AWS Device Farm.

Administrator layanan — Jika Anda bertanggung jawab atas sumber daya Device Farm di perusahaan Anda, Anda mungkin memiliki akses penuh ke Device Farm. Tugas Anda adalah menentukan fitur dan sumber daya Device Farm mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang cara perusahaan Anda dapat menggunakan IAM dengan Device Farm, lihatCara AWS Device Farm bekerja dengan IAM.

Administrator IAM — Jika Anda administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Device Farm. Untuk melihat contoh kebijakan berbasis identitas Device Farm yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas AWS Device Farm

Mengautentikasi dengan identitas

Autentikasi adalah cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Anda harus terautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengambil peran IAM.

Anda dapat masuk ke AWS sebagai identitas terfederasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. Pengguna AWS IAM Identity Center Pengguna (Pusat Identitas IAM), autentikasi Single Sign-On perusahaan Anda, dan kredensial Google atau Facebook Anda

merupakan contoh identitas terfederasi. Saat Anda masuk sebagai identitas gabungan, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil suatu peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal akses AWS. Untuk informasi selengkapnya tentang cara masuk ke AWS, lihat Cara masuk ke Akun AWS dalam Panduan Pengguna AWS Sign-In.

Jika Anda mengakses AWS secara terprogram, AWS memberikan Kit Pengembangan Perangkat Lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan peralatan AWS, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang cara menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan API AWS dalam Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Sebagai contoh, AWS menyarankan Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS dalam Panduan Pengguna IAM.

Pengguna root Akun AWS

Ketika membuat Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses penuh ke semua Layanan AWS dan sumber daya di akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk menggunakan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar tugas lengkap yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya andalkan kredensial temporer, dan bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, sebaiknya rotasikan kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci

akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran tersebut dimaksudkan untuk dapat diambil oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM merupakan identitas dalam Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara dalam AWS Management Console dengan berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL kustom. Untuk informasi selengkapnya tentang metode untuk menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas gabungan, Anda dapat membuat peran dan menentukan izin untuk peran tersebut. Saat identitas terfederasi diautentikasi, identitas tersebut dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi sekumpulan izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengaitkan izin yang ditetapkan ke peran dalam IAM. Untuk informasi tentang rangkaian izin, lihat Rangkaian izin dalam Panduan Pengguna AWS IAM Identity Center.
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) dengan akun berbeda untuk mengakses sumber daya yang ada

di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, pada beberapa Layanan AWS, Anda dapat menyertakan kebijakan secara langsung ke sumber daya (bukan menggunakan peran sebagai proksi). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.

- Akses lintas layanan – Sebagian Layanan AWS menggunakan fitur di Layanan AWS lainnya. Contoh, ketika Anda melakukan panggilan dalam layanan, umumnya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses maju (FAS) – Ketika Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan di AWS, Anda akan dianggap sebagai seorang pengguna utama. Saat menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya diajukan saat layanan menerima permintaan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Meneruskan sesi akses.
  - Peran IAM – Peran layanan adalah peran IAM yang diambil layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran terkait layanan – Peran terkait layanan adalah tipe peran layanan yang terkait dengan Layanan AWS. Layanan tersebut dapat mengambil peran untuk melakukan sebuah tindakan atas nama Anda. Peran terkait layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan peran IAM untuk mengelola kredensial sementara untuk aplikasi yang berjalan di instans EC2 dan mengajukan permintaan API AWS CLI atau AWS. Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan peran AWS ke instans EC2 dan menyediakannya bagi semua aplikasinya, Anda dapat membuat profil instans yang dilampirkan ke instans tersebut. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara.

Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Cara AWS Device Farm bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Device Farm, Anda harus memahami fitur IAM mana yang tersedia untuk digunakan dengan Device Farm. Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Device Farm dan AWS layanan lainnya dengan IAM, lihat AWSLayanan yang Bekerja dengan IAM di Panduan Pengguna IAM.

Topik

- Kebijakan berbasis identitas Device Farm
- Kebijakan berbasis sumber daya Device Farm
- Daftar kontrol akses
- Otorisasi berdasarkan tag Device Farm
- Peran IAM Device Farm

Kebijakan berbasis identitas Device Farm

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak dan ketentuan di mana tindakan tersebut diperbolehkan atau ditolak. Device Farm mendukung tindakan, sumber daya, dan kunci kondisi tertentu. Untuk mempelajari semua elemen yang Anda gunakan dalam kebijakan JSON, lihat Referensi Elemen Kebijakan JSON IAM dalam Panduan Pengguna IAM.

Tindakan

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama seperti operasi API AWS terkait. Ada beberapa pengecualian, misalnya tindakan

hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin melakukan operasi terkait.

Tindakan kebijakan di Device Farm menggunakan awalan berikut sebelum tindakan:`devicefarm:`. Misalnya, untuk memberikan izin kepada seseorang untuk memulai sesi Selenium dengan operasi `CreateTestGridUrl` API pengujian browser desktop Device Farm, Anda menyertakan `devicefarm:CreateTestGridUrl` tindakan tersebut dalam kebijakan. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. Device Farm mendefinisikan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan dengan koma seperti berikut:

```
"Action": [
      "devicefarm:action1",
      "devicefarm:action2"
```

Anda dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `List`, sertakan tindakan berikut:

```
"Action": "devicefarm:List*"
```

Untuk melihat daftar tindakan Device Farm, lihat Tindakan yang ditentukan oleh AWS Device Farm dalam Referensi Otorisasi Layanan IAM.

Sumber daya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek atau beberapa objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Sumber daya instans Amazon EC2 memiliki ARN berikut:

```
arn:${Partition}:ec2:${Region}:${Account}:instance/${InstanceId}
```

Untuk informasi lebih lanjut tentang format ARN, lihat Amazon Resource Name (ARN) dan Namespace Layanan AWS.

Misalnya, untuk menentukan instans `i-1234567890abcdef0` dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0"
```

Untuk menentukan semua instance milik akun, gunakan wildcard (*):

```
"Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*"
```

Beberapa tindakan Device Farm, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya. Dalam kasus tersebut, Anda harus menggunakan wildcard (*).

```
"Resource": "*"
```

Banyak tindakan API Amazon EC2 yang melibatkan beberapa sumber daya. Sebagai contoh, `AttachVolume` melampirkan volume Amazon EBS pada instans, sehingga pengguna IAM harus memiliki izin untuk menggunakan volume dan instans tersebut. Untuk menentukan beberapa sumber daya dalam satu pernyataan, pisahkan ARN dengan koma.

```
"Resource": [
      "resource1",
      "resource2"
```

Untuk melihat daftar tipe sumber daya Device Farm dan ARNnya, lihat Jenis sumber daya yang ditentukan oleh AWS Device Farm dalam Referensi Otorisasi Layanan IAM. Untuk mempelajari

tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh AWS Device Farm dalam Referensi Otorisasi Layanan IAM.

Kunci syarat

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, pengguna utama mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) memungkinkan Anda menentukan kondisi di mana suatu pernyataan akan diterapkan. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi kondisional yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam satu pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya dengan menggunakan operasi `AND` logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tanda yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tanda di Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi spesifik layanan. Untuk melihat semua kunci kondisi global AWS, lihat kunci konteks kondisi global AWS dalam Panduan Pengguna IAM.

Device Farm mendefinisikan rangkaian kunci kondisinya sendiri dan juga mendukung penggunaan beberapa kunci kondisi global. Untuk melihat semua kunci syarat global AWS, lihat Kunci Konteks Syarat Global AWS dalam Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Device Farm, lihat Kunci kondisi untuk AWS Device Farm Referensi Otorisasi Layanan IAM. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh AWS Device Farm dalam Referensi Otorisasi Layanan IAM.

Contoh-contoh

Untuk melihat contoh kebijakan berbasis identitas Device Farm, lihat. Contoh kebijakan berbasis identitas AWS Device Farm

Kebijakan berbasis sumber daya Device Farm

Device Farm tidak mendukung kebijakan berbasis sumber daya.

Daftar kontrol akses

Device Farm tidak mendukung daftar kontrol akses (ACL).

Otorisasi berdasarkan tag Device Farm

Anda dapat melampirkan tag ke sumber daya Device Farm atau meneruskan tag dalam permintaan ke Device Farm. Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tanda di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang menandai sumber daya Device Farm, lihatMenandai di Device Farm.

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Melihat proyek pengujian browser desktop Device Farm berdasarkan tag.

Peran IAM Device Farm

Peran IAM adalah entitas di AWS akun Anda yang memiliki izin tertentu.

Menggunakan kredensyal sementara dengan Device Farm

Device Farm mendukung penggunaan kredensyal sementara.

Anda dapat menggunakan kredensyal sementara untuk masuk dengan federasi untuk mengambil peran IAM atau peran lintas akun. Anda memperoleh kredensi keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRoleatau. GetFederationToken

Peran terkait layanan

Peran terkait layanan mengizinkan layanan AWS untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit, izin untuk peran terkait layanan.

Device Farm menggunakan peran terkait layanan dalam fitur pengujian browser desktop Device Farm. Untuk informasi tentang peran ini, lihat Menggunakan Peran Tertaut Layanan dalam pengujian browser desktop Device Farm dalam panduan pengembang.

Peran layanan

Device Farm tidak mendukung peran layanan.

Fitur ini memungkinkan layanan untuk menerima peran layanan atas nama Anda. Peran ini mengizinkan layanan untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun tersebut. Ini berarti administrator IAM dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

Mengelola akses menggunakan kebijakan

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan adalah objek di AWS yang, ketika terkait dengan identitas atau sumber daya, akan menentukan izinnya. AWS mengevaluasi kebijakan-kebijakan tersebut ketika seorang pengguna utama (pengguna, pengguna root, atau sesi peran) mengajukan permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, silakan lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses terhadap apa. Artinya, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat menjalankan peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk operasi. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau API AWS.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini

mengontrol jenis tindakan yang dapat dilakukan pengguna dan peran, di sumber daya mana, dan dengan ketentuan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Tabel berikut menguraikan kebijakan terkelola Device Farm AWS.

Perubahan	Deskripsi	Tanggal
AWSDeviceFarmFullAccess	Menyediakan akses penuh ke semua operasi AWS Device Farm.	Juli 15, 2015
AWSServiceRoleForDeviceFarmTestGrid	Memungkinkan Device Farm mengakses sumber daya AWS atas nama Anda.	20 Mei 2021

Jenis kebijakan lainnya

AWS mendukung jenis kebijakan tambahan yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda berdasarkan tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan di mana Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan secara eksplisit terhadap salah satu kebijakan ini akan mengesampingkan izin tersebut. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations

adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS yang dimiliki bisnis Anda secara terpusat. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda teruskan sebagai parameter saat Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit di salah satu kebijakan ini akan membatalkan izin tersebut. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Jika beberapa jenis kebijakan diberlakukan untuk satu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan ketika ada beberapa jenis kebijakan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas AWS Device Farm

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya Device Farm. Mereka juga tidak dapat melakukan tugas menggunakan API AWS Management Console, AWS CLI, or AWS. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat Kebijakan pada Tab JSON dalam Panduan Pengguna IAM.

Topik

- Praktik terbaik kebijakan
- Izinkan pengguna melihat izin mereka sendiri
- Mengakses satu proyek pengujian browser desktop Device Farm
- Melihat proyek pengujian browser desktop Device Farm berdasarkan tag

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Device Farm di akun Anda. Tindakan ini dikenai biaya untuk Akun AWS Anda. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulai menggunakan kebijakan yang dikelola AWS dan beralih ke izin dengan hak akses paling rendah – Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan yang dikelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Kebijakan ini ada di Akun AWS Anda. Sebaiknya Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan yang dikelola AWS atau kebijakan yang dikelola AWS untuk fungsi pekerjaan di Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukan ini dengan menentukan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan harus dikirim menggunakan SSL. Anda juga dapat menggunakan kondisi untuk memberi akses ke tindakan layanan jika digunakan melalui Layanan AWS yang spesifk, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Menggunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan berfungsi – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat validasi kebijakan Analizer Akses IAM di Panduan Pengguna IAM.
- Wajibkan autentikasi multi-faktor (MFA) – Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Akun AWS Anda, aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda.

Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Izinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan para pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau menggunakan AWS CLI atau AWS API secara terprogram.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

Mengakses satu proyek pengujian browser desktop Device Farm

Dalam contoh ini, Anda ingin memberikan pengguna IAM di AWS akun Anda akses ke salah satu proyek pengujian browser Device Farm Anda,. `arn:aws:devicefarm:us-west-2:111122223333:testgrid-project:123e4567-e89b-12d3-a456-426655441111` Anda ingin akun dapat melihat item yang terkait dengan proyek.

Selain `devicefarm:GetTestGridProject` titik akhir, akun harus memiliki,`devicefarm:ListTestGridSessions`, `devicefarm:GetTestGridSessiondevicefarm:ListTestGridSessionActions`, dan titik `devicefarm:ListTestGridSessionArtifacts` akhir.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"GetTestGridProject",
         "Effect":"Allow",
         "Action":[
            "devicefarm:GetTestGridProject"
         ],
         "Resource":"arn:aws:devicefarm:us-west-2:111122223333:testgrid-
project:123e4567-e89b-12d3-a456-426655441111"
      },
      {
         "Sid":"ViewProjectInfo",
         "Effect":"Allow",
         "Action":[
            "devicefarm:ListTestGridSessions",
            "devicefarm:ListTestGridSessionActions",
            "devicefarm:ListTestGridSessionArtifacts"
         ],
         "Resource":"arn:aws:devicefarm:us-west-2:111122223333:testgrid-*:123e4567-
e89b-12d3-a456-426655441111/*"
      }
   ]
}
```

Jika Anda menggunakan sistem CI, Anda harus memberikan setiap kredensyal akses unik pelari CI. Misalnya, sistem CI tidak mungkin membutuhkan lebih banyak izin daripada `devicefarm:ScheduleRun` atau`devicefarm:CreateUpload`. Kebijakan IAM berikut menguraikan kebijakan minimal untuk memungkinkan pelari CI memulai pengujian pengujian aplikasi bawaan Device Farm baru dengan membuat unggahan dan menggunakannya untuk menjadwalkan uji coba:

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "$id":"scheduleTestRuns",
         "effect":"Allow",
         "Action": [ "devicefarm:CreateUpload","devicefarm:ScheduleRun" ],
         "Resource": [
            "arn:aws:devicefarm:us-west-2:111122223333:project:123e4567-e89b-12d3-
a456-426655440000",
            "arn:aws:devicefarm:us-west-2:111122223333:*:123e4567-e89b-12d3-
a456-426655440000/*",
            ]
         }
    ]
}
```

Melihat proyek pengujian browser desktop Device Farm berdasarkan tag

Anda dapat menggunakan kondisi dalam kebijakan berbasis identitas untuk mengontrol akses ke sumber daya Device Farm berdasarkan tag. Contoh ini menunjukkan cara membuat kebijakan yang memungkinkan penayangan proyek dan sesi. Izin diberikan jika `Owner` tag sumber daya yang diminta cocok dengan nama pengguna akun yang meminta.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListTestGridProjectSessions",
            "Effect": "Allow",
            "Action": [
              "devicefarm:ListTestGridSession*",
```

```
              "devicefarm:GetTestGridSession",
              "devicefarm:ListTestGridProjects"
              ],
            "Resource": [
               "arn:aws:devicefarm:us-west-2:testgrid-project:*/*"
               "arn:aws:devicefarm:us-west-2:testgrid-session:*/*"
               ],
            "Condition": {
               "StringEquals": {"aws:TagKey/Owner":"${aws:username}"}
            }
        }
    ]
}
```

Anda dapat melampirkan kebijakan ini ke pengguna IAM di akun Anda. Jika pengguna bernama `richard-roe` mencoba melihat proyek atau sesi Device Farm, proyek harus diberi tag `0wner=richard-roe` atau`owner=richard-roe`. Jika tidak, pengguna ditolak aksesnya. Kunci tag kondisi `0wner` cocok dengan keduanya `0wner` dan `owner` karena nama kunci kondisi tidak peka huruf besar/kecil. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM JSON: Syarat dalam Panduan Pengguna IAM.

Memecahkan masalah identitas dan akses AWS Device Farm

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Device Farm dan IAM.

Saya tidak berwenang untuk melakukan tindakan di Device Farm

Jika Anda menerima kesalahan dalam AWS Management Console yang mengatakan Anda tidak berwenang untuk melakukan tindakan, Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna IAM,`mateojackson`, mencoba menggunakan konsol untuk melihat detail tentang proses, tetapi tidak memiliki `devicefarm:GetRun` izin.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 devicefarm:GetRun on resource: arn:aws:devicefarm:us-west-2:123456789101:run:123e4567-
e89b-12d3-a456-426655440000/123e4567-e89b-12d3-a456-426655441111
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk memungkinkannya mengakses `arn:aws:devicefarm:us-`

`west-2:123456789101:run:123e4567-e89b-12d3-a456-426655440000/123e4567-e89b-12d3-a456-426655441111` sumber daya `devicefarm:GetRun` pada menggunakan `devicefarm:GetRun` tindakan.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Device Farm.

Sebagian Layanan AWS mengizinkan Anda untuk memberikan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait-layanan. Untuk melakukan tindakan tersebut, Anda harus memiliki izin untuk memberikan peran pada layanan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Device Farm. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin melihat access key saya

Setelah membuat access key pengguna IAM, Anda dapat melihat access key ID Anda setiap saat. Namun, Anda tidak dapat melihat secret access key Anda lagi. Jika Anda kehilangan secret key, Anda harus membuat pasangan access key baru.

Access key terdiri dari dua bagian: access key ID (misalnya, `AKIAIOSFODNN7EXAMPLE`) dan secret access key (misalnya, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Seperti nama pengguna dan kata sandi, Anda harus menggunakan access key ID dan secret access key sekaligus untuk mengautentikasi permintaan Anda. Kelola access key Anda seaman nama pengguna dan kata sandi Anda.

> **Important**
>
> Jangan memberikan access key Anda kepada pihak ke tiga, bahkan untuk membantu menemukan ID pengguna kanonis Anda. Dengan melakukan ini, Anda mungkin memberi seseorang akses permanen ke AndaAkun AWS.

Saat Anda membuat pasangan access key, Anda diminta menyimpan access key ID dan secret access key di lokasi yang aman. secret access key hanya tersedia saat Anda membuatnya. Jika Anda kehilangan secret access key Anda, Anda harus menambahkan access key baru ke pengguna IAM Anda. Anda dapat memiliki maksimum dua access key. Jika Anda sudah memiliki dua, Anda harus menghapus satu pasangan kunci sebelum membuat pasangan baru. Untuk melihat instruksi, lihat Mengelola access keys di Panduan Pengguna IAM.

Saya seorang administrator dan ingin mengizinkan orang lain mengakses Device Farm

Untuk mengizinkan orang lain mengakses Device Farm, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang memerlukan akses. Mereka akan menggunakan kredensial untuk entitas tersebut untuk mengakses AWS. Anda kemudian harus melampirkan kebijakan ke entitas yang memberi mereka izin yang benar di Device Farm.

Untuk segera mulai, lihat Membuat pengguna dan grup khusus IAM pertama Anda di Panduan Pengguna IAM.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya Device Farm saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau pengguna di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi pengguna akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Device Farm mendukung fitur-fitur ini, lihatCara AWS Device Farm bekerja dengan IAM.

- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke pihak ketiga Akun AWS, lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara peran IAM dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Validasi kepatuhan untuk AWS Device Farm

Auditor pihak ketiga menilai keamanan dan kepatuhan AWS Device Farm sebagai bagian dari beberapa program kepatuhan AWS. Ini termasuk SOC, PCI, FedRAMP, HIPAA, dan lainnya.AWS Device Farmtidak dalam lingkup apapunAWSprogram kepatuhan.

Untuk daftar layanan AWS dalam cakupan program kepatuhan tertentu, lihat Layanan AWS dalam Cakupan Program Kepatuhan. Untuk informasi umum, lihat Program Kepatuhan AWS.

Anda bisa mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Pengunduhan Laporan dalam AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Device Farm ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.AWSmenyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Quick Start Keamanan dan Kepatuhan – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah untuk deployment lingkungan dasar yang fokus pada keamanan dan kepatuhan di AWS.
- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan Developer AWS Config – AWS Config menilai seberapa patuh konfigurasi sumber daya Anda terhadap praktik internal, panduan industri, dan peraturan.

- AWS Security Hub – Layanan AWS ini memberikan pandangan yang komprehensif tentang status keamanan Anda di dalam AWS yang membantu Anda memeriksa kepatuhan terhadap standar industri dan praktik terbaik yang terkait dengan keamanan.

Perlindungan data di AWS Device Farm

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS Device Farm (Device Farm). Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk memelihara kendali atas isi yang dihost pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya lindungi kredensial Akun AWS dan siapkan untuk masing-masing pengguna AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya AWS. Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pengelogan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama semua kontrol keamanan bawaan dalam Layanan AWS.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Device Farm atau lainnya Layanan AWS menggunakan konsol, APIAWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam

tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Enkripsi dalam bergerak

Titik akhir Device Farm hanya mendukung permintaan HTTPS (SSL/TLS) yang ditandatangani kecuali jika dinyatakan lain. Semua konten yang diambil dari atau ditempatkan di Amazon S3 melalui URL unggahan dienkripsi menggunakan SSL/TLS. Untuk informasi selengkapnya tentang cara permintaan HTTPS masukAWS, lihat Menandatangani permintaan AWS API di Referensi AWS Umum.

Merupakan tanggung jawab Anda untuk mengenkripsi dan mengamankan komunikasi apa pun yang dibuat oleh aplikasi Anda yang diuji dan aplikasi apa pun yang diinstal dalam proses menjalankan pengujian di perangkat.

Enkripsi diam

Fitur pengujian browser desktop Device Farm mendukung enkripsi saat istirahat untuk artefak yang dihasilkan selama pengujian.

Data pengujian perangkat seluler fisik Device Farm tidak dienkripsi saat istirahat.

Retensi data

Data di Device Farm disimpan untuk waktu yang terbatas. Setelah periode retensi berakhir, data akan dihapus dari penyimpanan dukungan Device Farm, tetapi metadata apa pun (ARN, tanggal upload, nama file, dan sebagainya) dipertahankan untuk penggunaan di masa mendatang. Tabel berikut mencantumkan periode retensi untuk berbagai jenis konten.

Jenis konten	Periode retensi (hari)
Aplikasi yang diunggah	30
Paket uji yang diunggah	30
Log	400

Jenis konten	Periode retensi (hari)
Rekaman video dan artefak lainnya	400

Anda bertanggung jawab untuk mengarsipkan konten apa pun yang ingin Anda simpan untuk waktu yang lebih lama.

Pengelolaan data

Data di Device Farm dikelola secara berbeda tergantung pada fitur mana yang digunakan. Bagian ini menjelaskan bagaimana data dikelola saat dan setelah Anda menggunakan Device Farm.

Pengujian browser desktop

Contoh yang digunakan selama sesi Selenium tidak disimpan. Semua data yang dihasilkan sebagai hasil dari interaksi browser dibuang ketika sesi berakhir.

Fitur ini saat ini mendukung enkripsi saat istirahat untuk artefak yang dihasilkan selama pengujian.

Pengujian perangkat fisik

Bagian berikut memberikan informasi tentang langkah-langkah yang AWS diperlukan untuk membersihkan atau menghancurkan perangkat setelah Anda menggunakan Device Farm.

Data pengujian perangkat seluler fisik Device Farm tidak dienkripsi saat istirahat.

Armada perangkat publik

Setelah eksekusi pengujian selesai, Device Farm melakukan serangkaian tugas pembersihan di setiap perangkat dalam armada perangkat publik, termasuk penghapusan instalasi aplikasi Anda. Jika kami tidak dapat memverifikasi penghapusan instalasi aplikasi Anda atau salah satu langkah pembersihan lainnya, perangkat akan menerima reset pabrik sebelum digunakan kembali.

> **Note**
>
> Data dapat bertahan di antara sesi dalam beberapa kasus, terutama jika Anda menggunakan sistem perangkat di luar konteks aplikasi Anda. Untuk alasan ini, dan karena Device Farm menangkap video dan log aktivitas yang terjadi selama Anda menggunakan setiap perangkat, kami menyarankan Anda untuk tidak memasukkan informasi sensitif (misalnya, akun Google

atau ID Apple), informasi pribadi, dan detail sensitif keamanan lainnya selama sesi pengujian otomatis dan akses jarak jauh Anda.

Perangkat pribadi

Setelah kedaluwarsa atau penghentian kontrak perangkat pribadi Anda, perangkat dihapus dari penggunaan dan dihancurkan dengan aman sesuai dengan kebijakan penghancuran AWS. Untuk informasi selengkapnya, lihat Bekerja dengan perangkat pribadi di AWS Device Farm.

Manajemen kunci

Saat ini, Device Farm tidak menawarkan manajemen kunci eksternal untuk enkripsi data, saat istirahat atau dalam perjalanan.

Privasi lalu lintas jaringan internet

Device Farm dapat dikonfigurasi, hanya untuk perangkat pribadi, untuk menggunakan titik akhir Amazon VPC untuk terhubung ke sumber daya Anda. AWS Akses ke AWS infrastruktur non-publik apa pun yang terkait dengan akun Anda (misalnya, instans Amazon EC2 tanpa alamat IP publik) harus menggunakan titik akhir Amazon VPC. Terlepas dari konfigurasi titik akhir VPC, Device Farm mengisolasi lalu lintas Anda dari pengguna lain di seluruh jaringan Device Farm.

Koneksi Anda di luar AWS jaringan tidak dijamin aman atau aman, dan Anda bertanggung jawab untuk mengamankan koneksi internet apa pun yang dibuat aplikasi Anda.

Ketahanan di AWS Device Farm

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Availability Zone lebih tersedia, memiliki toleransi kesalahan, dan dapat diskalakan dibandingkan dengan satu atau beberapa infrastruktur pusat data tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat AWS Infrastruktur Global.

Karena Device Farm tersedia di`us-west-2`Hanya wilayah, kami sangat menyarankan Anda menerapkan proses pencadangan dan pemulihan. Device Farm tidak boleh menjadi satu-satunya sumber konten yang diunggah.

Device Farm tidak menjamin ketersediaan perangkat publik. Perangkat ini dibawa masuk dan keluar dari kumpulan perangkat publik tergantung pada berbagai faktor, seperti tingkat kegagalan dan status karantina. Kami tidak menyarankan Anda bergantung pada ketersediaan satu perangkat di kolam perangkat publik.

Keamanan infrastruktur dalam AWS Device Farm

Sebagai layanan yang dikelola,AWS Device Farmdilindungi olehAWSkeamanan jaringan global. Untuk informasi tentangAWSLayanan keamanan dan bagaimanaAWSmelindungi infrastruktur, lihatAWSKeamanan Cloud. Untuk mendesain AndaAWSlingkungan menggunakan praktik terbaik untuk keamanan infrastruktur, lihatPerlindungan InfrastrukturdiPilar KeamananAWSKerangka Kerja yang Diarsiteksikan dengan Baik.

Anda menggunakanAWSmempublikasikan panggilan API untuk mengakses Device Farm melalui jaringan. Klien harus mendukung hal-hal berikut:

- Keamanan Lapisan Transportasi (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Cipher suite dengan perfect forward secrecy (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Keamanan infrastruktur untuk pengujian perangkat fisik

Perangkat dipisahkan secara fisik selama pengujian perangkat fisik. Isolasi jaringan mencegah komunikasi lintas perangkat melalui jaringan nirkabel.

Perangkat publik dibagikan, dan Device Farm melakukan upaya terbaik untuk menjaga keamanan perangkat dari waktu ke waktu. Tindakan tertentu, seperti upaya untuk memperoleh hak administrator lengkap pada perangkat (praktik yang disebut sebagairootingataujailbreaking), menyebabkan

perangkat publik dikarantina. Mereka dihapus dari kolam umum secara otomatis dan ditempatkan ke dalam tinjauan manual.

Perangkat pribadi hanya dapat diakses olehAWSakun yang secara eksplisit diizinkan untuk melakukannya. Device Farm secara fisik mengisolasi perangkat ini dari perangkat lain dan menyimpannya di jaringan terpisah.

Pada perangkat yang dikelola secara pribadi, pengujian dapat dikonfigurasi untuk menggunakan titik akhir Amazon VPC untuk mengamankan koneksi masuk dan keluar dariAWSakun.

Keamanan infrastruktur untuk pengujian browser desktop

Saat Anda menggunakan fitur pengujian browser desktop, semua sesi pengujian dipisahkan satu sama lain. Contoh selenium tidak dapat berkomunikasi silang tanpa pihak ketiga perantara, di luarAWS.

Semua lalu lintas ke SeleniumWebDriverpengontrol harus dilakukan melalui titik akhir HTTPS yang dihasilkan dengan`createTestGridUrl`.

Fitur pengujian browser desktop tidak mendukung konfigurasi titik akhir Amazon VPC saat ini. Anda bertanggung jawab untuk memastikan bahwa setiap instance pengujian Device Farm memiliki akses aman ke sumber daya yang diuji.

Analisis dan manajemen kerentanan konfigurasi di Device Farm

Device Farm memungkinkan Anda menjalankan perangkat lunak yang tidak dipelihara atau ditambal secara aktif oleh vendor, seperti vendor OS, vendor perangkat keras, atau operator telepon. Device Farm melakukan upaya terbaik untuk mempertahankan perangkat lunak terbaru, tetapi tidak menjamin bahwa versi perangkat lunak tertentu pada perangkat fisik adalah yang terbaru, dengan desain yang memungkinkan perangkat lunak yang berpotensi rentan untuk digunakan.

Misalnya, jika pengujian dilakukan pada perangkat yang menjalankan Android 4.4.2, Device Farm tidak menjamin bahwa perangkat telah ditambal terhadap perangkatkerentanan di Android dikenal sebagaiStageFright. Terserah vendor (dan terkadang operator) perangkat untuk memberikan pembaruan keamanan ke perangkat. Aplikasi berbahaya yang menggunakan kerentanan ini tidak dijamin akan ditangkap oleh karantina otomatis kami.

Perangkat pribadi dikelola sesuai perjanjian Anda denganAWS.

Device Farm melakukan upaya dengan itikad terbaik untuk mencegah aplikasi pelanggan dari tindakan sepertirootingataujailbreaking. Device Farm menghapus perangkat yang dikarantina dari kolam umum hingga ditinjau secara manual.

Anda bertanggung jawab untuk menjaga setiap pustaka atau versi perangkat lunak yang Anda gunakan dalam pengujian Anda, seperti roda Python dan permata Ruby, up to date. Device Farm menyarankan agar Anda memperbarui pustaka pengujian.

Sumber daya ini dapat membantu menjaga dependensi pengujian Anda tetap mutakhir:

- Untuk informasi tentang cara mengamankan permata Ruby, lihatPraktik KeamananpadaRubyGemssitus web.
- Untuk informasi tentang paket keamanan yang digunakan oleh Pipenv dan didukung oleh Python Packaging Authority untuk memindai grafik ketergantungan Anda untuk mengetahui kerentanan yang diketahui, lihatDeteksi Kerentanan Keamanandi atasGitHub.
- Untuk informasi tentang pemeriksa ketergantungan Maven Open Web Application Security Project (OWASP), lihatOTAWONDependencyCheckdi situs web OWASP.

Penting untuk diingat bahwa meskipun sistem otomatis tidak percaya ada masalah keamanan yang diketahui, itu tidak berarti bahwa tidak ada masalah keamanan. Selalu gunakan uji tuntas saat menggunakan perpustakaan atau alat dari pihak ketiga dan verifikasi tanda tangan kriptografi bila memungkinkan atau masuk akal.

Respon insiden di Device Farm

Device Farm terus memantau perangkat untuk perilaku yang mungkin mengindikasikan masalah keamanan. JikaAWSdibuat sadar akan kasus di mana data pelanggan, seperti hasil tes atau file yang ditulis ke perangkat publik, dapat diakses oleh pelanggan lain,AWSkontak pelanggan yang terpengaruh, sesuai dengan kebijakan peringatan dan pelaporan insiden standar yang digunakan di seluruhAWSlayanan.

Pencatatan dan pemantauan di Device Farm

Layanan ini mendukungAWS CloudTrail, yang merupakan layanan yang mencatatAWSpanggilan untuk AndaAkun AWSdan mengirimkan file log ke bucket Amazon S3. Dengan menggunakan informasi yang dikumpulkan olehCloudTrail, Anda dapat menentukan permintaan apa yang berhasil dibuatLayanan AWS, siapa yang membuat permintaan, kapan dibuat, dan sebagainya. Untuk

mempelajari lebih lanjut tentangCloudTrail, termasuk cara menyalakannya dan menemukan file log Anda, lihatAWS CloudTrailPanduan Pengguna.

Untuk informasi tentang penggunaanCloudTraildengan Device Farm, lihatMencatat panggilan AWS Device Farm API denganAWS CloudTrail.

Praktik terbaik keamanan untuk Device Farm

Device Farm menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik ini mungkin tidak sesuai atau cukup untuk lingkungan Anda, anggap praktik terbaik tersebut sebagai pertimbangan yang membantu dan bukan sebagai rekomendasi.

- Berikan sistem integrasi berkelanjutan (CI) apa pun yang Anda gunakan sesedikit mungkin hak istimewa di bawah IAM. Pertimbangkan untuk menggunakan kredensil sementara untuk setiap pengujian sistem CI sehingga meskipun sistem CI dikompromikan, itu tidak dapat membuat permintaan palsu. Untuk informasi selengkapnya tentang kredensi sementara, lihatPanduan Pengguna IAM.
- Gunakanadbperintah di lingkungan pengujian khusus untuk membersihkan konten apa pun yang dibuat oleh aplikasi Anda. Untuk informasi selengkapnya tentang lingkungan pengujian kustom, lihatBekerja dengan lingkungan pengujian khusus.

Batas di AWS Device Farm

Daftar berikut menjelaskan batas AWS Device Farm saat ini:

- Ukuran file maksimum aplikasi yang dapat Anda unggah adalah 4 GB.
- Tidak ada batasan jumlah perangkat yang dapat Anda sertakan dalam uji coba. Namun, jumlah maksimum perangkat yang akan diuji oleh Device Farm secara bersamaan selama uji coba adalah lima. (Jumlah ini dapat ditingkatkan berdasarkan permintaan.)
- Tidak ada batasan jumlah lari yang dapat Anda jadwalkan.
- Ada batas 150 menit untuk durasi sesi akses jarak jauh.
- Ada batas 150 menit untuk durasi uji coba otomatis.
- Jumlah maksimum pekerjaan dalam penerbangan, termasuk pekerjaan antrian yang tertunda di seluruh akun Anda, adalah 250. Ini adalah batas lunak.
- Tidak ada batasan jumlah perangkat yang dapat Anda sertakan dalam uji coba. Jumlah perangkat, atau pekerjaan, di mana Anda dapat menjalankan tes secara paralel pada waktu tertentu sama dengan konkurensi tingkat akun Anda. Konkurensi tingkat akun default untuk penggunaan terukur di AWS Device Farm adalah 5. Anda dapat meminta peningkatan jumlah ini hingga ambang batas tertentu tergantung pada kasus penggunaan. Konkurensi tingkat akun default untuk penggunaan yang tidak diukur sama dengan jumlah slot yang Anda berlangganan untuk platform itu.

Alat dan plugin untuk AWS Device Farm

Bagian ini berisi tautan dan informasi tentang bekerja dengan alat dan plugin AWS Device Farm. Anda dapat menemukan plugin Device Farm diAWS Labs diGitHub.

Jika Anda adalah pengembang Android, kami juga memilikiAplikasi sampel AWS Device Farm untuk Android diGitHub. Anda dapat menggunakan aplikasi dan contoh pengujian sebagai referensi untuk skrip pengujian Device Farm Anda sendiri.

Topik

- Integrasi AWS Device Farm dengan plugin Jenkins CI
- Plugin AWS Device Farm Gradle

Integrasi AWS Device Farm dengan plugin Jenkins CI

Plugin ini menyediakan fungsionalitas AWS Device Farm dari server integrasi berkelanjutan (CI) Jenkins Anda sendiri. Untuk informasi lebih lanjut, lihatJenkins (perangkat lunak).

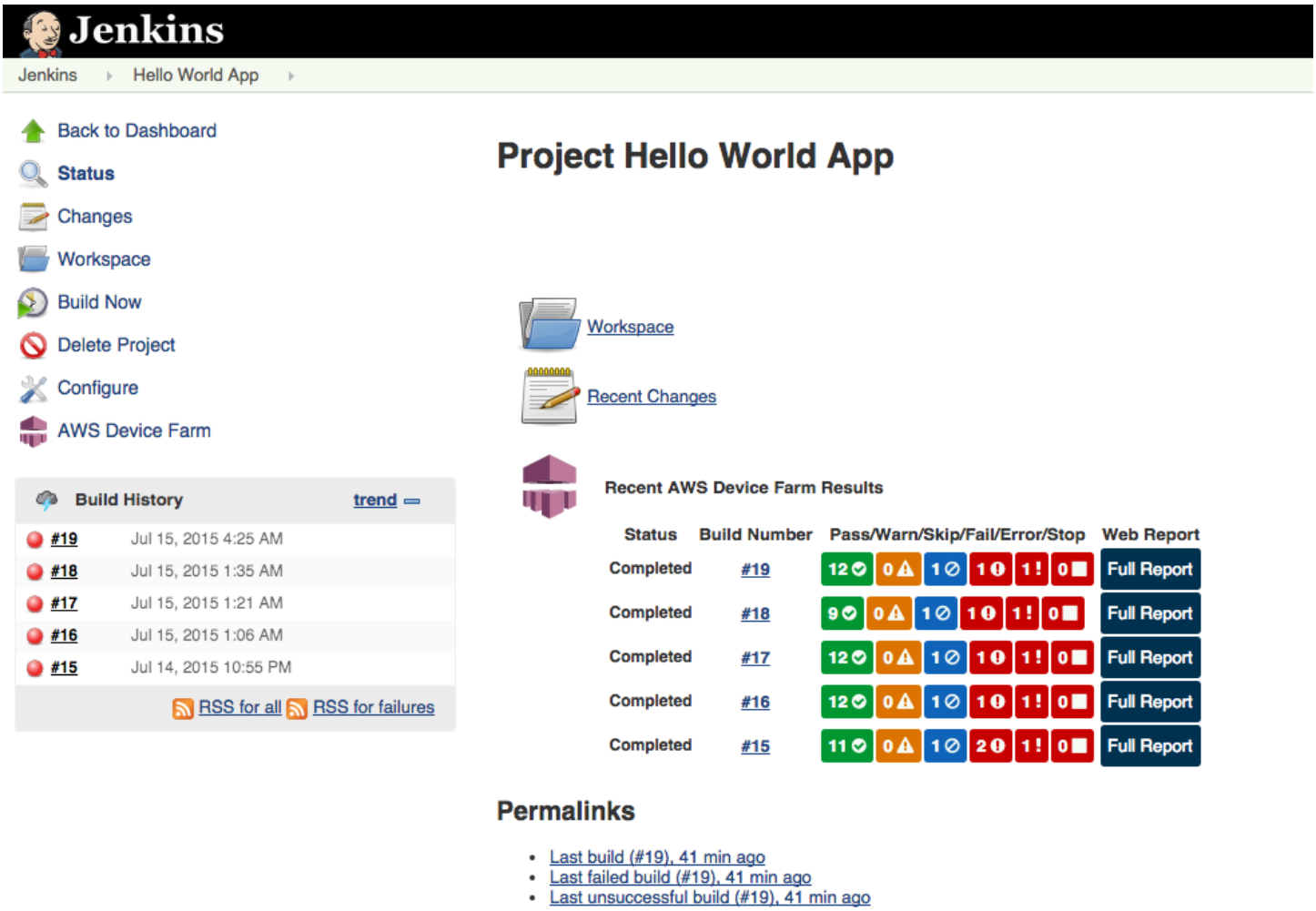
> **Note**
>
> Untuk mengunduh plugin Jenkins, bukaGitHubdan ikuti instruksi diLangkah 1: Menginstal plugin.

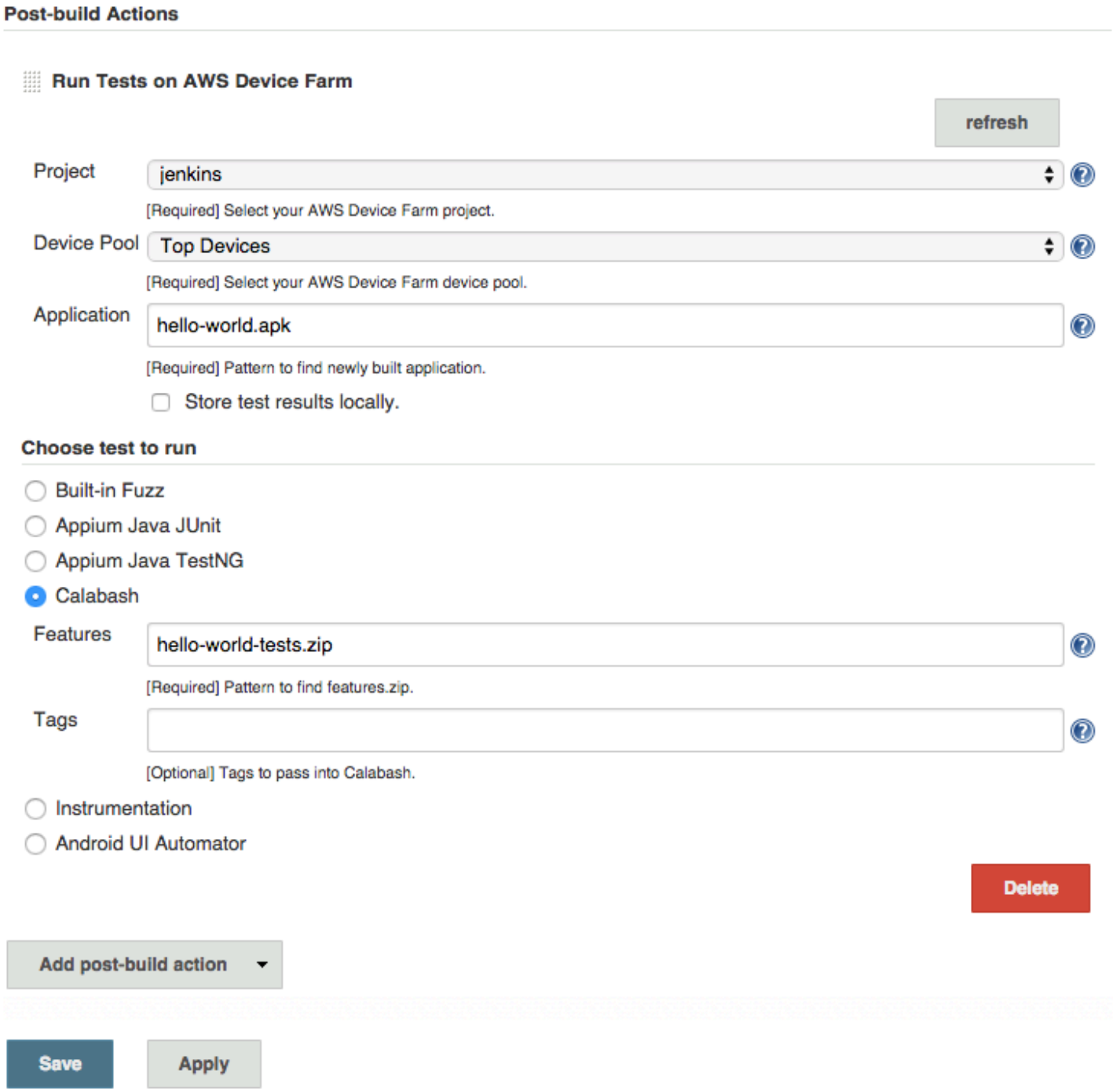
Bagian ini berisi serangkaian prosedur untuk menyiapkan dan menggunakan plugin Jenkins CI dengan AWS Device Farm.

Topik

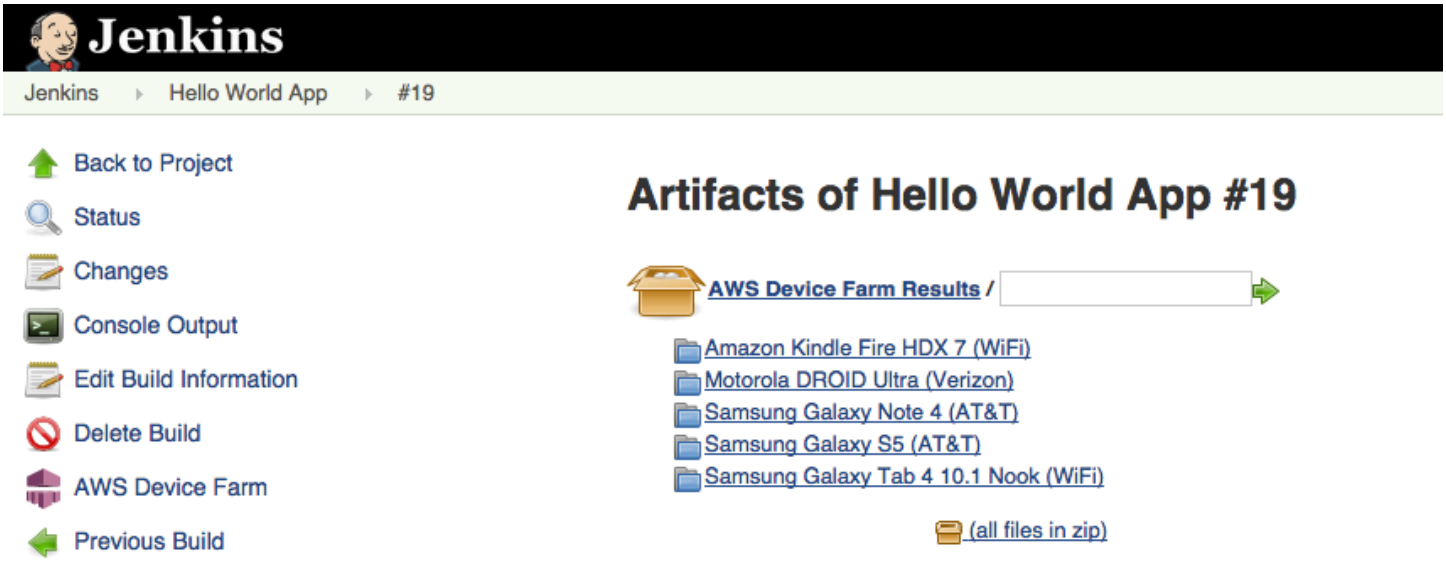
- Langkah 1: Menginstal plugin
- Langkah 2: MembuatAWS Identity and Access Managementpengguna untuk Plugin Jenkins CI Anda
- Langkah 3: Instruksi konfigurasi pertama kali
- Langkah 4: Menggunakan plugin dalam pekerjaan Jenkins
- Dependensi

Gambar-gambar berikut menunjukkan fitur plugin Jenkins CI.





Plugin ini juga dapat menarik semua artefak pengujian (log, tangkapan layar, dll.) Secara lokal:



Langkah 1: Menginstal plugin

Ada dua opsi untuk menginstal plugin Jenkins continuous integration (CI) untuk AWS Device Farm. Anda dapat mencari plugin dari dalamPlugin yang tersediadialog di Jenkins Web UI, atau Anda dapat men-downloadhpifile dan instal dari dalam Jenkins.

Instal dari dalam UI Jenkins

1. Temukan plugin dalam UI Jenkins dengan memilihKelola Jenkins,Kelola Plugin, dan kemudian pilihTersedia.
2. Cari aws-device-farm.
3. Instal plugin AWS Device Farm.
4. Pastikan plugin tersebut dimiliki olehJenkinspengguna.
5. Mulai ulang Jenkins.

Unduh plugin

1. Unduhhpiberkas langsung darihttp://updates.jenkins-ci.org/latest/aws-device-farm.hpi.
2. Pastikan plugin tersebut dimiliki olehJenkinspengguna.
3. Instal plugin menggunakan salah satu opsi berikut:

- Unggah plugin dengan memilihKelola Jenkins,Kelola Plugin,Lanjutan, dan kemudian pilihUnggah plugin.
- Tempatkanhpiberkas di direktori plugin Jenkins (biasanya`/var/lib/jenkins/plugins`).

4. Mulai ulang Jenkins.

Langkah 2: MembuatAWS Identity and Access Managementpengguna untuk Plugin Jenkins CI Anda

Kami menyarankan Anda untuk tidak menggunakanAWSakun root untuk mengakses Device Farm. Sebaliknya, buat yang baruAWS Identity and Access Management(IAM) pengguna (atau menggunakan pengguna IAM yang ada) diAWSakun, dan kemudian akses Device Farm dengan pengguna IAM tersebut.

Untuk membuat pengguna IAM baru, lihatMembuat Pengguna IAM (AWS Management Console). Pastikan untuk membuat kunci akses untuk setiap pengguna dan mengunduh atau menyimpan kredensi keamanan pengguna. Anda akan membutuhkan kredensialnya nanti.

Berikan izin kepada pengguna IAM untuk mengakses Device Farm

Untuk memberikan izin kepada pengguna IAM untuk mengakses Device Farm, buat kebijakan akses baru di IAM, lalu tetapkan kebijakan akses ke pengguna IAM sebagai berikut.

> **Note**
>
> TheAWSakun root atau pengguna IAM yang Anda gunakan untuk menyelesaikan langkah-langkah berikut harus memiliki izin untuk membuat kebijakan IAM berikut dan melampirkannya ke pengguna IAM. Untuk informasi lebih lanjut, lihatBekerja dengan Kebijakan

Untuk membuat kebijakan akses di IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Policies (Kebijakan).
3. Pilih Buat Kebijakan. (Jika tombol Memulai muncul, pilihlah, lalu pilih Buat Kebijakan.)
4. Di sebelah Buat Kebijakan Anda Sendiri, pilih Pilih.

5. UntukNama Kebijakan, ketik nama untuk kebijakan (misalnya,**`AWSDeviceFarmAccessPolicy`**).
6. UntukDeskripsi, ketik deskripsi yang membantu Anda mengaitkan pengguna IAM ini dengan proyek Jenkins Anda.
7. UntukDokumen Kebijakan, ketik pernyataan berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeviceFarmAll",
            "Effect": "Allow",
            "Action": [ "devicefarm:*" ],
            "Resource": [ "*" ]
        }
    ]
}
```

8. Pilih Buat Kebijakan.

Untuk menetapkan kebijakan akses ke pengguna IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Pengguna.
3. Pilih pengguna IAM kepada siapa Anda akan menetapkan kebijakan akses.
4. DiIzindaerah, untukKebijakan Terkelola, pilihLampirkan Kebijakan.
5. Pilih kebijakan yang baru saja Anda buat (misalnya,AWSDeviceFarmAccessPolicy).
6. Pilih Lampirkan Kebijakan.

Langkah 3: Instruksi konfigurasi pertama kali

Pertama kali Anda menjalankan server Jenkins Anda, Anda perlu mengkonfigurasi sistem sebagai berikut.

> **Note**
>
> Jika Anda menggunakanslot perangkat, fitur slot perangkat dinonaktifkan secara default.

1. Masuk ke antarmuka pengguna Web Jenkins Anda.
2. Di sisi kiri layar, pilihKelola Jenkins.
3. PilihKonfigurasikan Sistem.
4. Gulir ke bawah kePeternakan Perangkat AWSsundulan.
5. Salin kredensi keamanan Anda dariLangkah 2: Buat pengguna IAMdan tempelkan ID Kunci Akses dan Kunci Akses Rahasia Anda ke dalam kotak masing-masing.
6. Pilih Simpan.

Langkah 4: Menggunakan plugin dalam pekerjaan Jenkins

Setelah Anda menginstal plugin Jenkins, ikuti petunjuk ini untuk menggunakan plugin dalam pekerjaan Jenkins.

1. Masuk ke UI web Jenkins Anda.
2. Klik pekerjaan yang ingin Anda edit.
3. Di sisi kiri layar, pilihKonfigurasikan.
4. Gulir ke bawah keTindakan Pasca-bangunsundulan.
5. KlikTambahkan tindakan pasca-builddan pilihJalankan Pengujian di AWS Device Farm.
6. Pilih proyek yang ingin Anda gunakan.
7. Pilih kumpulan perangkat yang ingin Anda gunakan.
8. Pilih apakah Anda ingin artefak pengujian (seperti log dan tangkapan layar) diarsipkan secara lokal.
9. DiAplikasi, isi jalur ke aplikasi yang dikompilasi Anda.
10. Pilih tes yang ingin Anda jalankan dan isi semua bidang yang diperlukan.
11. Pilih Simpan.

Dependensi

Plugin Jenkins CI membutuhkanAWSMobile SDK 1.10.5 atau yang lebih baru. Untuk informasi selengkapnya dan untuk menginstal SDK, lihatSDK Seluler AWS.

Plugin AWS Device Farm Gradle

Plugin ini menyediakan integrasi AWS Device Farm dengan sistem build Gradle di Android Studio. Untuk informasi lebih lanjut, lihatGradle.

> **Note**
>
> Untuk mengunduh plugin Gradle, bukaGitHubdan ikuti instruksi diMembangun plugin Device Farm Gradle.

Plugin Device Farm Gradle menyediakan fungsionalitas Device Farm dari lingkungan Android Studio Anda. Anda dapat memulai tes pada ponsel dan tablet Android nyata yang dihosting oleh Device Farm.

Bagian ini berisi serangkaian prosedur untuk menyiapkan dan menggunakan Plugin Device Farm Gradle.

Topik

Langkah 1: Membangun plugin AWS Device Farm Gradle

Plugin ini menyediakan integrasi AWS Device Farm dengan sistem build Gradle di Android Studio. Untuk informasi lebih lanjut, lihatGradle.

> **Note**
>
> Membangun plugin adalah opsional. Plugin ini diterbitkan melalui Maven Central. Jika Anda ingin mengizinkan Gradle mengunduh plugin secara langsung, lewati langkah ini dan lanjutkan keLangkah 2: Menyiapkan plugin AWS Device Farm Gradle.

Untuk membangun plugin

1. Pergi keGitHubdan mengkloning repositori.
2. Membangun plugin menggunakan`gradle install`.

   Plugin diinstal ke repositori maven lokal Anda.

Langkah selanjutnya: Langkah 2: Menyiapkan plugin AWS Device Farm Gradle

Langkah 2: Menyiapkan plugin AWS Device Farm Gradle

Jika Anda belum melakukannya, kloning repositori dan instal plugin menggunakan prosedur di sini:Membangun plugin Device Farm Gradle.

Untuk mengonfigurasi Plugin AWS Device Farm Gradle

1. Tambahkan artefak plugin ke daftar ketergantungan Anda di`build.gradle`.

```
    buildscript {

        repositories {
            mavenLocal()
            mavenCentral()
        }

        dependencies {
            classpath 'com.android.tools.build:gradle:1.3.0'
            classpath 'com.amazonaws:aws-devicefarm-gradle-plugin:1.0'
        }
    }
```

2. Konfigurasikan plugin di`build.gradle`berkas. Konfigurasi khusus pengujian berikut harus berfungsi sebagai panduan Anda:

```
apply plugin: 'devicefarm'

devicefarm {

    // Required. The project must already exist. You can create a project in the
 AWS Device Farm console.
    projectName "My Project" // required: Must already exist.
```

```
   // Optional. Defaults to "Top Devices"
   // devicePool "My Device Pool Name"

   // Optional. Default is 150 minutes
   // executionTimeoutMinutes 150

   // Optional. Set to "off" if you want to disable device video recording during
a run. Default is "on"
   // videoRecording "on"

   // Optional. Set to "off" if you want to disable device performance monitoring
during a run. Default is "on"
   // performanceMonitoring "on"

   // Optional. Add this if you have a subscription and want to use your unmetered
slots
   // useUnmeteredDevices()

   // Required. You must specify either accessKey and secretKey OR roleArn.
roleArn takes precedence.
   authentication {
       accessKey "AKIAIOSFODNN7EXAMPLE"
       secretKey "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"

       // OR

       roleArn "arn:aws:iam::111122223333:role/DeviceFarmRole"
   }

   // Optionally, you can
   // - enable or disable Wi-Fi, Bluetooth, GPS, NFC radios
   // - set the GPS coordinates
   // - specify files and applications that must be on the device when your test
runs
   devicestate {
       // Extra files to include on the device.
       // extraDataZipFile file("path/to/zip")

       // Other applications that must be installed in addition to yours.
       // auxiliaryApps files(file("path/to/app"), file("path/to/app2"))

       // By default, Wi-Fi, Bluetooth, GPS, and NFC are turned on.
       // wifi "off"
```

```
        // bluetooth "off"
        // gps "off"
        // nfc "off"

        // You can specify GPS location. By default, this location is 47.6204,
 -122.3491
        // latitude 44.97005
        // longitude -93.28872
    }

    // By default, the Instrumentation test is used.
    // If you want to use a different test type, configure it here.
    // You can set only one test type (for example, Calabash, Fuzz, and so on)

    // Fuzz
    // fuzz { }

    // Calabash
    // calabash { tests file("path-to-features.zip") }

}
```

3. Jalankan pengujian Device Farm menggunakan tugas berikut:`gradle devicefarmUpload`.

   Output build akan mencetak tautan ke konsol Device Farm tempat Anda dapat memantau eksekusi pengujian.

Langkah selanjutnya: Menghasilkan pengguna IAM

Langkah 3: Menghasilkan pengguna IAM

AWS Identity and Access Management(IAM) membantu Anda mengelola izin dan kebijakan untuk bekerja denganAWSsumber daya. Topik ini memandu Anda dalam menghasilkan pengguna IAM dengan izin untuk mengakses sumber daya AWS Device Farm.

Jika Anda belum melakukannya, selesaikan langkah 1 dan 2 sebelum membuat pengguna IAM.

Kami menyarankan Anda untuk tidak menggunakanAWSakun root untuk mengakses Device Farm. Sebagai gantinya, buat pengguna IAM baru (atau gunakan pengguna IAM yang ada) diAWSakun, dan kemudian akses Device Farm dengan pengguna IAM tersebut.

> **Note**
>
> TheAWSakun root atau pengguna IAM yang Anda gunakan untuk menyelesaikan langkah-langkah berikut harus memiliki izin untuk membuat kebijakan IAM berikut dan melampirkannya ke pengguna IAM. Untuk informasi lebih lanjut, lihatBekerja dengan Kebijakan.

Untuk membuat pengguna baru dengan kebijakan akses yang tepat di IAM

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Pengguna.
3. Pilih Buat Antrean Baru.
4. Masukkan nama pengguna pilihan Anda.

   Sebagai contoh, **`GradleUser`**.
5. Pilih Create (Buat).
6. Pilih Unduh Kredensial dan simpan di lokasi di mana Anda dapat dengan mudah mengambilnya nanti.
7. Pilih Tutup.
8. Pilih nama pengguna dalam daftar.
9. Di bawahIzin, perluasKebijakan Inlineheader dengan mengklik panah bawah di sebelah kanan.
10. PilihKlik di sinidi mana ia mengatakan,Tidak ada kebijakan inline untuk ditampilkan. Untuk membuatnya, klik di sini.
11. Pada layar Atur izin, pilih Kebijakan Kustom.
12. Pilih Pilih.
13. Beri nama kebijakan Anda, seperti **`AWSDeviceFarmGradlePolicy`**.
14. Tempel kebijakan berikut ke dalam Dokumen Kebijakan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DeviceFarmAll",
            "Effect": "Allow",
```

```
                "Action": [ "devicefarm:*" ],
                "Resource": [ "*" ]
            }
        ]
    }
```

15. Pilih Terapkan Kebijakan.

Langkah selanjutnya:Mengkonfigurasi jenis pengujian.

Untuk informasi lebih lanjut, lihatMembuat Pengguna IAM (AWS Management Console)atauMenyiapkan.

Langkah 4: Mengkonfigurasi jenis tes

Secara default, plugin AWS Device Farm Gradle menjalankan Bekerja dengan instrumentasi untuk Android dan AWS Device Farm pengujian. Jika Anda ingin menjalankan pengujian Anda sendiri atau menentukan parameter tambahan, Anda dapat memilih untuk mengonfigurasi jenis pengujian. Topik ini memberikan informasi tentang setiap jenis pengujian yang tersedia dan apa yang perlu Anda lakukan di Android Studio untuk mengonfigurasinya agar dapat digunakan. Untuk informasi selengkapnya tentang jenis pengujian yang tersedia di Device Farm, lihatBekerja dengan jenis pengujian di AWS Device Farm.

Jika Anda belum melakukannya, selesaikan langkah 1 — 3 sebelum mengonfigurasi jenis pengujian.

> **Note**
>
> Jika Anda menggunakan slot perangkat, fitur slot perangkat dinonaktifkan secara default.

Appium

Device Farm menyediakan dukungan untuk Appium Java JUnit dan TestNG untuk Android.

- Appium (di bawah Java (JUnit))
- Appium (di bawah Jawa (TestNG))

Anda dapat memilih `useTestNG()` atau`useJUnit()`. `JUnit`adalah default dan tidak perlu ditentukan secara eksplisit.

```
    appium {
        tests file("path to zip file") // required
        useTestNG() // or useJUnit()
    }
```

Bawaan: bulu halus

Device Farm menyediakan tipe uji fuzz bawaan, yang secara acak mengirimkan peristiwa antarmuka pengguna ke perangkat dan kemudian melaporkan hasilnya.

```
    fuzz {

       eventThrottle 50 // optional default
       eventCount 6000  // optional default
       randomizerSeed 1234 // optional default blank

     }
```

Untuk informasi selengkapnya, lihat Bawaan: fuzz (Android dan iOS).

Instrumentasi

Device Farm menyediakan dukungan untuk instrumentasi (JUnit, Espresso, Robotium, atau pengujian berbasis instrumen apa pun) untuk Android. Untuk informasi selengkapnya, lihat Bekerja dengan instrumentasi untuk Android dan AWS Device Farm.

Saat menjalankan pengujian instrumentasi di Gradle, Device Farm menggunakan `.apk` file yang dihasilkan dari direktori AndroidTest sebagai sumber pengujian Anda.

```
    instrumentation {

        filter "test filter per developer docs" // optional

    }
```

Dependensi

Runtime

- Plugin Device Farm Gradle membutuhkanAWSMobile SDK 1.10.15 atau yang lebih baru. Untuk informasi selengkapnya dan untuk menginstal SDK, lihatSDK Seluler AWS.

- API uji pembuat alat Android 0.5.2
- Apache Commons Lang3 3.3.4

Untuk Tes Unit

- Testng 6.8.8
- Jmockit 1,19
- Alat gradle Android 1.3.0

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir panduan ini.

Perubahan	Deskripsi	Tanggal Diubah
Dukungan AL2	Device Farm sekarang mendukung lingkungan pengujian AL2 untuk Android. Pelajari lebih lanjut tentang AL2.	6 November 2023
Migrasi dari Standar ke lingkungan pengujian Kustom	Panduan migrasi yang diperbarui untuk mendokumentasikan penghentian pengujian mode standar pada Desember 2023.	September 3, 2023
Dukungan VPC ENI	Device Farm sekarang memungkinkan perangkat pribadi untuk menggunakan fitur konektivitas VPC-ENI untuk membantu pelanggan terhubung dengan aman ke titik akhir pribadi mereka yang dihosting di AWS, perangkat lunak lokal, atau penyedia cloud lainnya. Pelajari lebih lanjut tentang VPC-ENI.	15 Mei 2023
Pembaruan UI Polaris	Konsol Device Farm sekarang mendukung kerangka kerja Polaris.	28 Juli 2021
Dukungan Python 3	Device Farm sekarang mendukung Python 3 dalam pengujian mode kustom. Pelajari lebih lanjut tentang menggunakan Python 3 dalam paket pengujian Anda: • Appium (Python) • Appium (Python)	20 April 2020
Informasi keamanan baru dan informasi tentang AWS sumber daya penandaan.	Untuk membuat AWS layanan pengamanan lebih mudah dan lebih komprehensif, bagian baru tentang keamanan telah dibangun. Untuk membaca lebih lanjut, lihat Keamanan di AWS Device Farm	27 Maret 2020

Perubahan	Deskripsi	Tanggal Diubah
	Bagian baru tentang penandaan di Device Farm telah ditambahkan. Untuk mempelajari lebih lanjut tentang penandaan, lihatMenandai di Device Farm.	
Penghapusan Akses Perangkat Langsung.	Direct Device Access (debugging jarak jauh pada perangkat pribadi) tidak lagi tersedia untuk penggunaa n umum. Untuk pertanyaan tentang ketersediaan Direct Device Access di masa mendatang, silakan hubungi kami.	9 September 2019
Perbarui konfigurasi plugin Gradle	Konfigurasi plugin Gradle yang direvisi sekarang menyertak an versi konfigurasi gradle yang dapat disesuaikan, dengan parameter opsional yang dikomentari. Pelajari lebih lanjut tentang Menyiapkan plugin Device Farm Gradle.	16 Agustus 2019
Persyaratan baru untuk pengujian berjalan dengan XCTest	Untuk pengujian yang menggunakan framework XCTest, Device Farm sekarang memerlukan paket aplikasi yang dibuat untuk pengujian. Pelajari lebih lanjut tentang the section called “XCTest”.	4 Februari 2019
Support untuk jenis pengujian Appium Node.js dan Appium Ruby di lingkungan khusus	Anda sekarang dapat menjalankan pengujian di lingkunga n pengujian kustom Appium Node.js dan Appium Ruby. Pelajari lebih lanjut tentang Bekerja dengan jenis pengujian di AWS Device Farm.	Januari 10, 2019
Support untuk server Appium versi 1.7.2 di lingkungan standar dan kustom. Support untuk versi 1.8.1 menggunakan file YAMM spesifika si pengujian kustom di lingkungan pengujian khusus.	Sekarang Anda dapat menjalankan pengujian di lingkunga n pengujian standar dan kustom dengan server Appium versi 1.72, 1.71, dan 1.6.5. Anda juga dapat menjalank an pengujian dengan versi 1.8.1 dan 1.8.0 menggunak an file YAMM spesifikasi pengujian kustom di lingkunga n pengujian kustom. Pelajari lebih lanjut tentang Bekerja dengan jenis pengujian di AWS Device Farm.	2 Oktober 2018

Perubahan	Deskripsi	Tanggal Diubah
Lingkungan uji kustom	Dengan lingkungan pengujian khusus, Anda dapat memastikan pengujian berjalan seperti yang dilakukan di lingkungan lokal Anda. Device Farm sekarang menyediak an dukungan untuk live log dan streaming video, sehingga Anda bisa mendapatkan umpan balik instan tentang pengujian yang dijalankan di lingkungan pengujian khusus. Pelajari lebih lanjut tentang Bekerja dengan lingkungan pengujian khusus.	Agustus, 16 2018
Support untuk menggunakan Device Farm sebagai penyedia AWS CodePipeline pengujian	Anda sekarang dapat mengonfigurasi pipeline AWS CodePipeline untuk menggunakan AWS Device Farm berjalan sebagai tindakan pengujian dalam proses rilis Anda. CodePipeline memungkinkan Anda untuk dengan cepat menghubungkan repositori Anda untuk membangun dan menguji tahapan untuk mencapai sistem integrasi berkelanjutan yang disesuaikan dengan kebutuhan Anda. Pelajari lebih lanjut tentang Menggunakan AWS Device Farm diCodePipelinetahap uji.	Juli, 19 2018
Support untuk Perangkat Pribadi	Anda sekarang dapat menggunakan perangkat pribadi untuk menjadwalkan uji coba dan memulai sesi akses jarak jauh. Anda dapat mengelola profil dan pengaturan untuk perangkat ini, membuat titik akhir VPC Amazon untuk menguji aplikasi pribadi, dan membuat sesi debugging jarak jauh. Pelajari lebih lanjut tentang Bekerja dengan perangkat pribadi di AWS Device Farm.	2 Mei 2018
Support untuk Appium 1.6.3	Anda sekarang dapat mengatur versi Appium untuk pengujian kustom Appium Anda.	21 Maret 2017
Tetapkan batas waktu eksekusi untuk uji coba	Anda dapat mengatur batas waktu eksekusi untuk uji coba atau untuk semua pengujian dalam proyek. Pelajari lebih lanjut tentang Tetapkan batas waktu eksekusi untuk pengujian yang dijalankan di AWS Device Farm.	9 Februari 2017

Perubahan	Deskripsi	Tanggal Diubah
Pembentukan Jaringan	Anda sekarang dapat mensimulasikan koneksi jaringan dan kondisi untuk uji coba. Pelajari lebih lanjut tentang Simulasikan koneksi dan kondisi jaringan untuk AWS Device Farm berjalan.	8 Desember 2016
Bagian Pemecahan Masalah Baru	Sekarang Anda dapat memecahkan masalah unggahan paket pengujian menggunakan serangkaian prosedur yang dirancang untuk mengatasi pesan kesalahan yang mungkin Anda temui di konsol Device Farm. Pelajari lebih lanjut tentang Memecahkan masalah kesalahan Device Farm.	Agustus 10, 2016
Sesi Akses Jarak Jauh	Anda sekarang dapat mengakses dan berinteraksi dari jarak jauh dengan satu perangkat di konsol. Pelajari lebih lanjut tentang Bekerja dengan akses jarak jauh.	19 April 2016
Slots Perangkat Layanan Mandiri	Anda sekarang dapat membeli slot perangkat menggunak anAWS Management Console,AWS Command Line Interface, atau API. Pelajari lebih lanjut tentang cara Beli slot perangkat di Device Farm.	22 Maret 2016
Cara menghentikan uji coba	Anda sekarang dapat menghentikan pengujian berjalan menggunakanAWS Management Console, theAWS Command Line Interface, atau API. Pelajari lebih lanjut tentang cara Hentikan proses di AWS Device Farm.	22 Maret 2016
Jenis uji UI XCTest baru	Anda sekarang dapat menjalankan pengujian kustom XCTest UI pada aplikasi iOS. Pelajari lebih lanjut tentang jenis XCTest UI tes.	Maret 8, 2016
Jenis pengujian Appium Python baru	Anda sekarang dapat menjalankan pengujian kustom Appium Python di aplikasi Android, iOS, dan web. Pelajari lebih lanjut tentang Bekerja dengan jenis pengujian di AWS Device Farm.	19 Januari 2016

Perubahan	Deskripsi	Tanggal Diubah
Jenis pengujian Aplikasi Web	Anda sekarang dapat menjalankan Appium Java JUnit dan tes kustom TestNG pada aplikasi web. Pelajari lebih lanjut tentang Bekerja dengan pengujian aplikasi web di AWS Device Farm.	19 November 2015
Plugin Gradle AWS Device Farm	Pelajari lebih lanjut tentang cara menginstal dan menggunakanPlugin Perangkat Pertanian Gradle.	28 September 2015
Uji Bawaan Android Baru: Explorer	Pengujian explorer meng-crawl aplikasi Anda dengan menganalisis setiap layar seolah-olah itu adalah pengguna akhir dan mengambil tangkapan layar saat mengeksplorasi.	16 September 2015
Dukungan iOS ditambahkan	Pelajari lebih lanjut tentang menguji perangkat iOS dan menjalankan pengujian iOS (termasuk XCTest) di. Bekerja dengan jenis pengujian di AWS Device Farm	4 Agustus 2015
Rilis publik awal	Ini adalah rilis publik awal dari AWS Device Farm Developer Guide.	Juli 13, 2015

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.