Simian Army - Taller 8 MISO-4208

Chaos Monkey

Que es ?

Chaos Monkey es un servicio que identifica grupos de sistemas y termina aleatoriamente uno de los sistemas en un grupo de AWS. El servicio opera en un tiempo controlado (no se ejecuta los fines de semana y días festivos) y el intervalo (solo funciona durante el horario comercial). En la mayoría de los casos, hemos diseñado nuestras aplicaciones para que sigan funcionando cuando un compañero se desconecta, pero en esos casos especiales queremos asegurarnos de que haya personas disponibles para resolver y aprender de cualquier problema. Con esto en mente, Chaos Monkey solo funciona en horario comercial con la intención de que los ingenieros estén alertas y puedan responder.

Porque ejecutarlo ?

Las fallas ocurren, y ocurren inevitablemente cuando menos se desea. Si su aplicación no puede tolerar un fallo del sistema, ¿lo averiguaría si lo ubicaran a las 3 de la mañana o después de que ya se haya tomado el café de la mañana en la oficina? Incluso si está seguro de que su arquitectura puede tolerar una falla del sistema, ¿está seguro de que todavía podrá hacerlo la próxima semana? ¿Qué tal el próximo mes? El software es complejo y dinámico, esa "solución simple" que implementó la semana pasada podría tener consecuencias no deseadas. ¿Los equilibradores de carga de tráfico detectan y enrutan correctamente las solicitudes en caso de fallas del sistema? ¿Puedes reconstruir tus sistemas de manera confiable? Tal vez un ingeniero "parcheó rápidamente" un sistema en vivo la semana pasada y olvidó realizar los cambios en su repositorio de origen.

Estrategias de ataque

Originalmente, Netflix Chaos Monkey simplemente cerraría limpiamente una instancia a través de las API de EC2. Para simular más escenarios de fallas, ahora hay muchas maneras diferentes en que el mono del caos puede 'romper' una instancia, para simular diferentes tipos de fallas. Si su aplicación puede hacer frente a todos ellos, es más probable que pueda hacer frente a situaciones de error de "incógnitas desconocidas".

Después de que el mono del caos selecciona una instancia para su terminación, elige una estrategia de caos al azar, de la lista de estrategias de caos habilitadas. Puede habilitar /

deshabilitar una estrategia editando el archivo caos.properties. Además, algunas estrategias no son siempre aplicables; algunos pueden requerir acceso SSH o solo pueden aplicarse a instancias con volúmenes de EBS. Si la estrategia no es aplicable, no debe preocuparse, simplemente no será elegida.

Estas son las estrategias:

Shutdown instance *(Simius Mortus)*

Cierra la instancia usando la API EC2. La clásica estrategia del mono del caos.

Activar con: ***simianarmy.chaos.shutdowninstance.enabled = true***

Block all network traffic *(Simius Quies)*

Esto elimina todos los grupos de seguridad de la instancia y lo mueve a un grupo de seguridad que no permite ningún acceso. Por lo tanto, la instancia se está ejecutando, pero no se puede llegar a ella a través de la red. Esto solo puede funcionar en instancias de VPC.

Activar con: ***simianarmy.chaos.blockallnetworktraffic.enabled = true***

Opcionalmente configure el grupo de seguridad para usar: ***simianarmy.chaos.blockallnetworktraffic.group = blocked-network*** (blocked-network es el valor predeterminado)

El grupo de seguridad se crea automáticamente; si el grupo de seguridad ya existe, se usará tal cual. Entonces, si quiere dejar, por ejemplo el puerto 22 abierto para fines de diagnóstico, agréguelo al grupo de seguridad configurado.

Detach all EBS volumes *(Simius Amputa)*

Esto separa la fuerza todos los volúmenes de EBS de la instancia, simulando una falla de EBS. Por lo tanto, la instancia se está ejecutando, pero la E/S del disco EBS fallará.

Activar con: ***simianarmy.chaos.detachvolumes.enabled = true***
Advertencia: este mono puede causar la pérdida de datos, uselo con cuidado.

SSH monkeys

Los monos restantes funcionan ejecutando scripts en la instancia. Para configurar, debe establecer la clave SSH que se utilizará para iniciar sesión en la instancia.

SSH username: ***simianarmy.chaos.ssh.user = root*** (el usuario predeterminado es root)
SSH key path: ***simianarmy.chaos.ssh.key = ~/monkeyboy/.ssh/id_rsa*** (la ruta a su SSH key)

Burn-CPU *(Simius Cogitarius)*

Este mono ejecuta procesos intensivos de CPU, simulando un vecino ruidoso o una CPU defectuosa. La instancia tendrá efectivamente una CPU mucho más lenta.

Activar con: ***simianarmy.chaos.burncpu.enabled = true*** (Requiere que se configure SSH.)

Burn-IO *(Simius Occupatus)*

Este mono ejecuta procesos intensivos en disco, simulando un vecino ruidoso o un disco defectuoso. La instancia tendrá efectivamente un disco mucho más lento.

Activar con: ***simianarmy.chaos.burnio.enabled = true*** (Requiere que se configure SSH.)

Fill Disk *(Simius Plenus)*

Este mono escribe un archivo enorme en el dispositivo raíz, llenando el disco raíz EC2 (normalmente relativamente pequeño).

Activar con: ***simianarmy.chaos.filldisk.enabled = true*** (Requiere que se configure SSH.)

Kill Processes *(Simius Delirius)*

Este mono mata cualquier programa java o python que encuentre cada segundo, simulando una aplicación defectuosa, una instalación dañada o una instancia defectuosa. La instancia está bien, pero la aplicación java / python que se ejecuta en ella fallará continuamente.

Habilítelo con: ***simianarmy.chaos.killprocesses.enabled = true*** (Requiere que se configure SSH.)

Null-Route *(Simius Desertus)*

Este mono desenruta la red 10.0.0.0/8, que es utilizada por la red interna de EC2. Todo el tráfico de red EC2 <-> EC2 fallará.

Activar con: ***simianarmy.chaos.nullroute.enabled = true*** (Requiere que se configure SSH.)

Fail DNS *(Simius Nonomenius)*

Este mono usa iptables para bloquear el puerto 53 para TCP y UDP; esos son los puertos de tráfico DNS. Esto simula una falla de sus servidores DNS.

Activar con: ***simianarmy.chaos.faildns.enabled = true*** (Requiere que se configure SSH.)

Fail EC2 API *(Simius Noneccius)*

Este mono pone entradas de host ficticias en /etc/hosts, por lo que todas las comunicaciones de la API EC2 fallarán. Esto simula una falla del plano de control EC2. Por supuesto, si su aplicación no utiliza la API EC2 de la instancia, entonces no se verá afectada por completo.

Activar con: ***simianarmy.chaos.failec2.enabled = true*** (Requiere que se configure SSH.)

Fail S3 API *(Simius Amnesius)*

Este mono pone entradas de host ficticias en /etc/hosts, por lo que toda la comunicación S3 fallará. Esto simula una falla de S3. Por supuesto, si su aplicación no usa S3, entonces no se verá afectada por completo.

Activar con: ***simianarmy.chaos.fails3.enabled = true*** (Requiere que se configure SSH.)

Fail DynamoDB API *(Simius Nodynamus)*

Este mono pone entradas de host ficticias en /etc/hosts, por lo que todas las comunicaciones de DynamoDB fallarán. Esto simula una falla de DynamoDB. Al igual que con sus similares, esto solo afectará a las instancias que usen DynamoDB.

Activar con: ***simianarmy.chaos.faildynamodb.enabled = true*** (Requiere que se configure SSH.)

Network Corruption *(Simius Politicus)*

Este mono usa la API de conformación de tráfico para corromper una gran parte de los paquetes de red. Esto estimula la degradación de la red EC2.

Activar con: ***simianarmy.chaos.networkcorruption.enabled = true*** (Requiere que se configure SSH.)

Network Latency *(Simius Tardus)*

Este mono usa la API de conformación de tráfico para introducir latencia (1 segundo + - 50%) a todos los paquetes de red. Esto simula la degradación de la red EC2.

Habilitar con: ***simianarmy.chaos.networklatency.enabled = true*** (Requiere que se configure SSH.)

Network Loss *(Simius Perditus)*

Este mono usa la API de conformación de tráfico para eliminar una fracción de todos los paquetes de red. Esto simula la degradación de la red EC2.

Activar con: ***simianarmy.chaos.networkloss.enabled = true*** (Requiere que se configure SSH.)

Tomado de:
https://github.com/Netflix/SimianArmy/wiki/The-Chaos-Monkey-Army
https://github.com/Netflix/SimianArmy/wiki/Chaos-Monkey

Janitor Monkey

Que es ?

Es un servicio que se ejecuta en la nube de Amazon Web Services (AWS) en busca de recursos no utilizados para limpiar. El diseño de Janitor Monkey es lo suficientemente flexible como para permitir extenderlo a trabajar con otros proveedores de la nube y recursos de la nube. El servicio está configurado para ejecutarse, de forma predeterminada, en días de la semana no festivos a las 11 a. M. El programa puede ser fácilmente reconfigurado para adaptarse a las necesidades de su negocio.

Janitor Monkey determina si un recurso debe ser un candidato de limpieza aplicando un conjunto de reglas al respecto. Si alguna de las reglas determina que el recurso es un candidato de limpieza, Janitor Monkey marca el recurso y programa un tiempo para limpiarlo. El diseño de Janitor Monkey también hace que sea sencillo personalizar el conjunto de reglas o agregar nuevas.

Dado que siempre puede haber excepciones cuando desea mantener un recurso no utilizado más tiempo, antes de que Janitor Monkey elimine un recurso, el propietario del recurso recibirá una notificación un número configurable de días antes del tiempo de limpieza. Esto es para evitar que Janitor Monkey elimine un recurso que todavía necesita. El propietario del recurso puede marcar los recursos que desea mantener como excepciones y Janitor Monkey los dejará en paz.

Porque ejecutarlo ?

Una de las grandes ventajas de pasar de un centro de datos privado a la nube es que tiene acceso rápido y fácil a recursos nuevos casi ilimitados. Para enviar una nueva versión de la aplicación puede crear rápidamente un nuevo clúster, o cuando necesita más disco, solo adjunte un nuevo volumen, haga una copia de seguridad de sus datos, sólo haga una instantánea, pruebe una nueva idea, solo cree nuevas instancias y llegue a trabajo. La desventaja de esta flexibilidad es que es bastante fácil perder la pista de los recursos de la nube que ya no se necesitan. Tal vez se olvidó de eliminar el clúster con la versión anterior de su aplicación u olvidó destruir el volumen cuando ya no necesitaba el disco adicional. Tomar

instantáneas es ideal para copias de seguridad, pero ¿realmente las necesitas desde hace 12 meses? No es solo el olvido lo que puede causar problemas, por ejemplo, los errores de red pueden hacer que su solicitud para eliminar un volumen no utilizado se pierda.

A menudo hay recursos no utilizados que cuestan dinero a los usuarios de la nube, y necesitábamos una solución para rectificar este problema. Los ingenieros diligentes pueden eliminar manualmente los recursos no utilizados, pero necesitamos una forma de detectarlos y limpiarlos automáticamente. La solución es Janitor Monkey.

Cómo limpia el Janitor Monkey ?

Janitor Monkey trabaja en un proceso de "marcar, notificar y eliminar". Cuando Janitor Monkey marca un recurso como candidato de limpieza, programa un tiempo para eliminar el recurso. El tiempo de eliminación se especifica en la regla que marca el recurso. Todos los recursos están asociados con un correo electrónico del propietario, que se puede especificar como una etiqueta en el recurso o puede extender rápidamente Janitor Monkey para obtener la información de su sistema interno. La forma más simple es usar una dirección de correo electrónico predeterminada, p. la lista de correo electrónico de su equipo para todos los recursos.

Puede configurar un número de días para especificar cuándo dejar que Janitor Monkey envíe notificaciones al propietario del recurso antes de la finalización programada. Por defecto, el número es 3, lo que significa que el propietario recibirá una notificación 3 días hábiles antes de la fecha de finalización. Durante el período de 3 días, el propietario del recurso puede decidir si el recurso está bien para eliminarlo. En caso de que un recurso deba retenerse por más tiempo, el propietario puede usar una interfaz REST simple para indicar que el recurso no ha sido limpiado por Janitor Monkey. El propietario siempre puede usar otra interfaz REST para eliminar la bandera y Janitor Monkey podrá administrar el recurso nuevamente. Cuando Janitor Monkey ve un recurso marcado como candidato a limpieza y el tiempo de finalización programado ya ha pasado, eliminará el recurso. El propietario del recurso también puede eliminar el recurso manualmente si quiere liberar el recurso antes para ahorrar costos. Cuando el estado del recurso cambia, lo que hace que el recurso no sea un candidato de limpieza, p. un volumen separado de EBS se adjunta a una instancia, Janitor Monkey desmarca el recurso y no se producirá la terminación.

Reglas del Janitor Monkey

A continuación se describe cómo se establecen algunas reglas del janitor monkey

Regla de configuración de limpieza de instancias que no están en el grupo de escalado automático

Las siguientes propiedades se utilizan para configurar la regla utilizada para limpiar instancias huérfanas que no están en ningún grupo de escalado automático.

- **simianarmy.janitor.rule.orphanedInstanceRule.enabled**

  Esta propiedad especifica si Janitor monkey limpiará las instancias huérfanas. Si no desea eliminar instancias huérfanas, puede establecer la propiedad en false para deshabilitar la regla. El valor por defecto es true.

  simianarmy.janitor.rule.orphanedInstanceRule.enabled = true

- **simianarmy.janitor.rule.orphanedInstanceRule.instanceAgeThreshold**

  Una instancia huérfana se marca como candidata de limpieza si se ha lanzado durante más días que los especificados en esta propiedad. El valor predeterminado es 2, lo que significa que la instancia huérfana está marcada como candidata de limpieza y está programada para finalizar si se ha lanzado durante más de 2 días.

  simianarmy.janitor.rule.orphanedInstanceRule.instanceAgeThreshold = 2

- **simianarmy.janitor.rule.orphanedInstanceRule.retentionDaysWithOwner**

  Esta propiedad especifica el número de días hábiles que se conserva la instancia después de que se envía una notificación sobre la terminación cuando la instancia tiene un propietario. El valor predeterminado es 3.

  simianarmy.janitor.rule.orphanedInstanceRule.retentionDaysWithOwner = 3

- **simianarmy.janitor.rule.orphanedInstanceRule.retentionDaysWithoutOwner**

  Esta propiedad especifica el número de días hábiles que la instancia se conserva después de que se envía una notificación sobre la terminación cuando la instancia no tiene un propietario. El valor predeterminado es 8.
  simianarmy.janitor.rule.orphanedInstanceRule.retentionDaysWithoutOwner = 8

Establecer la regla de limpieza de volúmenes de EBS separados

Las siguientes propiedades se utilizan para configurar la regla utilizada para limpiar volúmenes que se han separado de instancias para ciertos días. Como AWS no proporciona una forma de

rastrear cuándo se separa un volumen de EBS, proporcionamos un mono acompañante llamado Mono de etiquetado de volumen para rastrear esta información. Puede modificar el archivo volumeTagging.properties para habilitarlo.

- **simianarmy.janitor.rule.oldDetachedVolumeRule.enabled**

  Esta propiedad especifica si Janitor monkey limpiará los volúmenes separados. Si no desea limpiar volúmenes separados, puede establecer la propiedad en false para deshabilitar la regla. El valor por defecto es true.

  simianarmy.janitor.rule.oldDetachedVolumeRule.enabled = true

- **simianarmy.janitor.rule.oldDetachedVolumeRule.detachDaysThreshold**

  Un volumen se considera un candidato de limpieza después de ser separado por el número de días especificado en esta propiedad. El valor predeterminado es 30.

  simianarmy.janitor.rule.oldDetachedVolumeRule.detachDaysThreshold = 30

- **simianarmy.janitor.rule.oldDetachedVolumeRule.retentionDays**

  Esta propiedad especifica el número de días hábiles que se retiene el volumen después de que se envía una notificación sobre la finalización. El valor predeterminado es 7.

  simianarmy.janitor.rule.oldDetachedVolumeRule.retentionDays = 7

Establecer regla de limpieza de instantáneas sin referencia

Las siguientes propiedades se utilizan para configurar la regla utilizada para limpiar instantáneas que no tienen imágenes generadas a partir de ellas y se lanzaron para ciertos días.

- **simianarmy.janitor.rule.noGeneratedAMIRule.enabled**

  Esta propiedad especifica si Janitor monkey limpiará instantáneas sin referencia. Puede establecer la propiedad en false para deshabilitar la regla. El valor por defecto es verdadero.

  simianarmy.janitor.rule.noGeneratedAMIRule.enabled = true

- **simianarmy.janitor.rule.noGeneratedAMIRule.ageThreshold**

Una instantánea sin referencia se considera un candidato de limpieza después del lanzamiento por el número de días especificado en esta propiedad. El valor predeterminado es 30.

simianarmy.janitor.rule.noGeneratedAMIRule.ageThreshold = 30

- **simianarmy.janitor.rule.noGeneratedAMIRule.retentionDays**

  Esta propiedad especifica el número de días hábiles que se conserva la instantánea después de que se envía una notificación sobre la finalización. El valor predeterminado es 7.

  simianarmy.janitor.rule.noGeneratedAMIRule.retentionDays = 7

- **simianarmy.janitor.rule.noGeneratedAMIRule.ownerEmail**

  Esta propiedad especifica un correo electrónico de propietario de reemplazo al limpiar instantáneas de EBS con esta regla. Esto permitirá que los correos electrónicos de notificación se envíen a un destino de correo electrónico diferente en lugar de al propietario del recurso. Esto podría ser útil para las organizaciones que crean una gran cantidad de instantáneas como parte de su proceso de compilación y no es necesario notificar a los propietarios.

  El valor predeterminado para esta propiedad es nulo, lo que significa que el propietario del recurso recibirá una notificación y no el valor de reemplazo.

  simianarmy.janitor.rule.noGeneratedAMIRule.ownerEmail = null

Configuración de la regla de limpieza de grupos de escalado automático vacíos

Las siguientes propiedades se utilizan para configurar la regla utilizada para limpiar grupos de escalado automático que no tienen instancias activas y la configuración de inicio tiene más de ciertos días.

- **simianarmy.janitor.Eureka.enabled**

  La propiedad a continuación especifica si Eureka/Discovery está o no disponible para que lo use Janitor monkey. Discovery/Eureka se utiliza en las reglas para limpiar grupos de escala automática para determinar si un grupo de escala automática tiene una instancia 'activa', es decir, una instancia que está registrada y arriba en Discovery/Eureka. Debería establecer esta propiedad en falso si no tiene

Discovery/Eureka configurado en su entorno. El valor por defecto de esta propiedad es "false".

simianarmy.janitor.Eureka.enabled = false

- **simianarmy.janitor.rule.oldEmptyASGRule.enabled**

  Esta propiedad especifica si Janitor monkey limpiará los grupos de escalado automático vacíos. Puede establecer la propiedad en false para deshabilitar la regla. El valor por defecto es true.

  simianarmy.janitor.rule.oldEmptyASGRule.enabled = true

- **simianarmy.janitor.rule.oldEmptyASGRule.launchConfigAgeThreshold**

  Un grupo de escala automática sin instancias activas se considera un candidato de limpieza cuando su configuración de inicio es anterior al número de días especificado en esta propiedad. El valor predeterminado es 50.

  simianarmy.janitor.rule.oldEmptyASGRule.launchConfigAgeThreshold = 50

- **simianarmy.janitor.rule.oldEmptyASGRule.retentionDays**

  The number of business days an empty auto-scaling group is retained after a notification is sent for the termination. The default value is 10.

  simianarmy.janitor.rule.oldEmptyASGRule.retentionDays = 10

Configuración de la regla de limpieza de grupos de escalado automático suspendidos

Las siguientes propiedades se utilizan para configurar la regla utilizada para limpiar grupos de escalado automático que no tienen instancias activas y se han suspendido del tráfico ELB asociado para ciertos días.

- **simianarmy.janitor.rule.suspendedASGRule.enabled**

  Esta propiedad especifica si Janitor monkey limpiará los grupos de autoescalamiento suspendidos. Puede establecer la propiedad en false para deshabilitar la regla. El valor por defecto es true.

simianarmy.janitor.rule.suspendedASGRule.enabled = true

- **simianarmy.janitor.rule.suspendedASGRule.suspensionAgeThreshold**

  Un grupo de escalamiento automático sin instancias activas se considera un candidato de limpieza cuando se ha suspendido del tráfico ELB asociado durante el número de días especificado en esta propiedad; el valor predeterminado es 2.

  simianarmy.janitor.rule.suspendedASGRule.suspensionAgeThreshold = 2

- **simianarmy.janitor.rule.suspendedASGRule.retentionDays**

  Esta propiedad especifica el número de días hábiles que el grupo de escalamiento automático se conserva después de que se envía una notificación para la finalización. El valor predeterminado es 5.

  simianarmy.janitor.rule.suspendedASGRule.retentionDays = 5

Establecer la regla de limpieza de configuraciones de inicio

La regla del conserje utiliza las siguientes propiedades para limpiar las configuraciones de inicio que no son utilizadas por ningún grupo o instancias de escalado automático y son anteriores a ciertos días.

- **simianarmy.janitor.rule.oldUnusedLaunchConfigRule.enabled**

  Esta propiedad especifica si Janitor monkey limpiará las configuraciones de inicio no utilizadas que tengan más de un día de antigüedad. Puede establecer la propiedad en false para deshabilitar la regla. El valor por defecto es true.

  simianarmy.janitor.rule.oldUnusedLaunchConfigRule.enabled = true

- **simianarmy.janitor.rule.oldUnusedLaunchConfigRule.ageThreshold**

  Una configuración de inicio no utilizada se considera un candidato de limpieza cuando es anterior a la cantidad de días especificada en la propiedad a continuación. El valor predeterminado es 4.

  simianarmy.janitor.rule.oldUnusedLaunchConfigRule.ageThreshold = 4

- **simianarmy.janitor.rule.oldUnusedLaunchConfigRule.retentionDays**

  Esta propiedad especifica el número de días hábiles que se guarda la configuración de inicio después de que se envía una notificación para la terminación. El valor predeterminado es 3.

  simianarmy.janitor.rule.oldUnusedLaunchConfigRule.retentionDays = 3

Establecer regla de limpieza de imágenes

Las siguientes propiedades son utilizadas por la regla del conserje para limpiar imágenes.

- **simianarmy.janitor.image.crawler.lookBackDays**

  La propiedad a continuación especifica el número de días para mirar hacia atrás en el historial al rastrear la última información de referencia de las imágenes. Por defecto miramos hacia atrás hasta 60 días.

  simianarmy.janitor.image.crawler.lookBackDays = 60

- **simianarmy.janitor.image.ownerId**

  La propiedad a continuación especifica la identificación del propietario que las imágenes tienen para ser administradas por Janitor Monkey. Si no se establece una ID de propietario, se devuelven todas las imágenes de la cuenta de AWS. Por defecto, la línea se comenta y no se establece ningún ID de propietario.

  #simianarmy.janitor.image.ownerId = 1234567890

- **simianarmy.janitor.rule.unusedImageRule.enabled**
  Esta regla es utilizada por la regla del conserje para limpiar las imágenes que no han sido utilizadas por ninguna instancia y las configuraciones de inicio, y no se han usado para crear otras imágenes, en los últimos días. Esta regla está deshabilitada por defecto, debe tener Edda ejecutándose y habilitado para usar esta regla, ya que el historial de la imagen es necesario para determinar la última vez que se hizo referencia a las imágenes.

  simianarmy.janitor.rule.unusedImageRule.enabled = false

- **simianarmy.janitor.rule.unusedImageRule.lastReferenceDaysThreshold**

Una imagen no utilizada se considera un candidato de limpieza cuando no se hace referencia a la cantidad de días especificada en la propiedad a continuación. El valor predeterminado es 45.

simianarmy.janitor.rule.unusedImageRule.lastReferenceDaysThreshold = 45

- **simianarmy.janitor.rule.unusedImageRule.retentionDays**

  La propiedad a continuación especifica el número de días hábiles que se guarda la imagen después de que se envía una notificación para la finalización. El valor predeterminado es 3.

  simianarmy.janitor.rule.unusedImageRule.retentionDays = 3

Tomado de:
https://github.com/Netflix/SimianArmy/wiki/Janitor-Settings
https://github.com/Netflix/SimianArmy/wiki/Janitor-Home

Conformity Monkey

Que es ?

Conformity Monkey es un servicio que se ejecuta en la nube de Amazon Web Services (AWS) buscando instancias que no se ajusten a las reglas predefinidas para las mejores prácticas. El diseño de Conformity Monkey es lo suficientemente flexible como para permitir extenderlo y trabajar con otros proveedores de nube y recursos de la nube. Por defecto, la verificación de conformidad se realiza cada hora. El programa puede ser fácilmente reconfigurado para adaptarse a las necesidades de su negocio.

Conformity Monkey determina si una instancia no es conforme aplicando un conjunto de reglas sobre ella. Si alguna de las reglas determina que la instancia no se está cumpliendo, el mono envía una notificación por correo electrónico al propietario de la instancia. Proporcionamos una colección de reglas de conformidad en la versión de fuente abierta que actualmente se usan en Netflix y que se consideran lo suficientemente generales como para ser utilizadas por la mayoría de los usuarios. El diseño de Conformity Monkey también facilita la personalización de reglas o la adición de nuevas.

Puede haber excepciones cuando desea ignorar las advertencias de una regla de conformidad específica para algunas aplicaciones. Por ejemplo, un grupo de seguridad para abrir un puerto específico probablemente no es necesario por instancias de algunas aplicaciones. Le permitimos personalizar el conjunto de reglas de conformidad que se aplicarán a un clúster de instancias excluyendo las innecesarias. O puede optar por excluirse por completo de un grupo específico de Conformity Monkey, por lo que no se envía ninguna notificación.

Porque ejecutarlo ?

La informática en la nube facilita mucho el lanzamiento de nuevas aplicaciones o el inicio de nuevas instancias. En Netflix, los ingenieros pueden lanzar fácilmente una nueva aplicación en Asgard con unos pocos clics. Con esta libertad, a veces hay consecuencias en las que las aplicaciones o instancias lanzadas pueden no seguir algunas de las mejores prácticas, tal vez el ingeniero no conocía las mejores prácticas o simplemente se olvidaba de ellas. Por ejemplo, algunos grupos de seguridad necesarios pueden faltar en las instancias y pueden causar lagunas de seguridad. O tal vez no se haya definido una url de verificación de estado para las instancias en Eureka, lo que daría como resultado la detección automática de fallas y la desactivación de failover.

Cómo funciona Conformity Monkey ?

Conformity Monkey funciona en dos etapas: marcar y notificar. Primero, Conformity Monkey recorre todos los grupos de escalado automático en su nube y aplica el conjunto especificado de reglas de conformidad a las instancias de cada grupo. Si alguna regla de conformidad determina que una instancia no se ajusta, el grupo de escalado automático se marca como no conforme y se registran las instancias que violan la regla. Cada grupo de escalado automático está asociado con un correo electrónico de propietario, que se puede obtener de un sistema interno o se puede configurar en un archivo de configuración. La forma más simple es usar una dirección de correo electrónico predeterminada, p. la lista de correo electrónico de su equipo para todos los grupos de autoescalamiento. Conformity Monkey envía al propietario una notificación por correo electrónico sobre los grupos no conformes, con los detalles de la regla de conformidad rota y las instancias que no superaron la verificación de conformidad. Los propietarios de la aplicación pueden entonces tomar las medidas necesarias para arreglar las instancias fallidas o excluir la regla de conformidad si creen que la verificación de conformidad no es necesaria para la aplicación. Le permitimos configurar diferentes frecuencias para la verificación de conformidad y la notificación. Por ejemplo, en Netflix, la verificación de conformidad se realiza cada hora, y la notificación solo se envía una vez al día al mediodía. Esto es para reducir la cantidad de correos electrónicos que las personas reciben sobre la misma advertencia de conformidad. El resultado en tiempo real de la verificación de conformidad para cada grupo de escalado automático se muestra en una IU separada.

Reglas del Conformity Monkey

A continuación se describen algunas reglas del conformity monkey

- **simianarmy.conformity.rule.SameZonesInElbAndAsg.enabled**

  Esta propiedad se utiliza para permitir que la regla de conformidad verifique si hay discrepancias en las zonas de disponibilidad entre cualquier grupo de escalado automático y sus equilibradores de carga elásticos en un clúster.

  simianarmy.conformity.rule.SameZonesInElbAndAsg.enabled = true

- **simianarmy.conformity.rule.InstanceInSecurityGroup.enabled**

  Esta propiedad se usa para permitir que la regla de conformidad verifique si todas las instancias de un clúster se encuentran en los grupos de seguridad necesarios.

  simianarmy.conformity.rule.InstanceInSecurityGroup.enabled = true

- **simianarmy.conformity.rule.InstanceInSecurityGroup.requiredSecurityGroups**

  Esta propiedad especifica los grupos de seguridad necesarios en la regla de conformidad InstanceInSecurityGroup.

  simianarmy.conformity.rule.InstanceInSecurityGroup.requiredSecurityGroups = foo, bar

- **simianarmy.conformity.rule.InstanceTooOld.enabled**

  Esta propiedad se utiliza para permitir que la regla de conformidad verifique si hay alguna instancia que sea anterior a ciertos días.

  simianarmy.conformity.rule.InstanceTooOld.enabled = true

- **simianarmy.conformity.rule.InstanceTooOld.instanceAgeThreshold**

  Esta propiedad especifica el número de días utilizados en InstanceInSecurityGroup, cualquier instancia que sea anterior a esta cantidad de días se considera no conforme.

  simianarmy.conformity.rule.InstanceTooOld.instanceAgeThreshold = 180

- **simianarmy.conformity.rule.InstanceInVPC.enabled**

  Esta propiedad se utiliza para permitir que la regla de conformidad verifique si sus instancias están en una nube privada virtual de Amazon (VPC).

  simianarmy.conformity.rule.InstanceInVPC.enabled = true

- **simianarmy.conformity.rule.InstanceHasStatusUrl.enabled**

  Esta propiedad se utiliza para permitir que la regla de conformidad verifique si todas las instancias en el clúster tienen una url de estado definida en Discovery/Eureka. La regla se agrega a Conformity Monkey sólo cuando Eureka también está habilitado.

  simianarmy.conformity.rule.InstanceHasStatusUrl.enabled = true

- **simianarmy.conformity.rule.InstanceHasHealthCheckUrl.enabled**

  Esta propiedad se utiliza para permitir que la regla de conformidad verifique si todas las instancias en el clúster tienen una url de verificación de estado definida en Discovery/Eureka. La regla se agrega a Conformity Monkey sólo cuando Eureka también está habilitado.

  simianarmy.conformity.rule.InstanceHasHealthCheckUrl.enabled = true

Tomado de:
https://github.com/Netflix/SimianArmy/wiki/Conformity-Settings
https://github.com/Netflix/SimianArmy/wiki/Conformity-Home

Gerson Uriel Florez Morales

gu.florez@uniandes.edu.co
201624374