Progettazione per l'alta disponibilità e la resilienza nelle applicazioni Amazon EKS

AWS Linee guida prescrittive

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Linee guida prescrittive: Progettazione per l'alta disponibilità e la resilienza nelle applicazioni Amazon EKS

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Introduzione 1
Progettazione dell'HA e della resilienza 2
Distribuisci i carichi di lavoro 2
Utilizza i vincoli di diffusione della topologia dei pod 3
Affinità e antiaffinità dei pod 7
Budget Pod per le interruzioni 9
Sonde e controlli sanitari 10
Sonda di avvio 10
Sonda di vivacità 10
Sonda di prontezza 11
Controlli dello stato delle risorse e del sistema di bilanciamento del carico in ingresso 11
Ganci per il ciclo di vita dei container 12
Comprendi lo sfratto dei pod durante le interruzioni zonali 14
Implementazione del trasferimento zonale di Amazon EKS per una migliore resilienza 14
Comprensione del meccanismo di spostamento zonale 15
Metodi di attivazione del turno zonale 15
Prerequisiti per un effettivo cambiamento di zona 16
Raccomandazioni per la resilienza alle perturbazioni zonali 16
Completamento e ripristino dei turni 17
Conclusioni 18
Resources 19
Cronologia dei documenti 20
..... xxi

Progettazione per disponibilità e resilienza elevate nelle applicazioni Amazon EKS

Haofei Feng, Frank Fan e Rus Kalakutskiy, Amazon Web Services ()AWS

Ottobre 2025 (storia del documento)

Garantire un'elevata disponibilità (HA) e resilienza nella progettazione delle applicazioni è fondamentale per raggiungere un obiettivo di ripristino (RPO) e un obiettivo di tempo di ripristino (RTO) vicini allo zero. Man mano che le organizzazioni migrano e modernizzano sempre più le proprie applicazioni negli ambienti Kubernetes, la richiesta di soluzioni robuste e scalabili continua ad aumentare. Amazon Elastic Kubernetes Service (Amazon EKS) ti aiuta a gestire in modo efficiente applicazioni containerizzate su larga scala.

Questa guida approfondisce una serie di raccomandazioni e best practice ampiamente riconosciute per la progettazione e la gestione di applicazioni di microservizi Amazon EKS. Basate su una vasta esperienza e implementazioni nel mondo reale, queste informazioni offrono una guida preziosa per architetti e sviluppatori. Implementa questi consigli per ottenere prestazioni, affidabilità e scalabilità elevate delle applicazioni basate su Kubernetes e ottenere operazioni affidabili.

Considerazioni sulla progettazione di elevata disponibilità e resilienza

Il modello di responsabilità condivisa diventa più complesso con Kubernetes. La disponibilità e la resilienza del piano di controllo di Amazon EKS sono gestite da Amazon Web Services (AWS). La tua organizzazione gestisce il piano dati, che può influire in modo significativo sulle prestazioni e sulla disponibilità delle tue applicazioni di microservizi.

Quando progetti un'applicazione ad alta disponibilità e resilienza su Amazon EKS, considera i seguenti componenti:

- L'applicazione per microservizi: i suoi pod e contenitori
- Il piano dati del carico di lavoro: Ingress Controller, pod, componenti di sistema come Amazon Virtual Private Cloud (Amazon VPC) Container Network Interface (CNI), sidecar service mesh e kube-proxy
- Il livello di gestione del carico di lavoro: controller, controller di ammissione, motori di policy di rete e archiviazione persistente dei dati per questi componenti
- Il piano di controllo di Kubernetes
- Infrastruttura: nodi, rete e dispositivi di rete

Per le prime tre considerazioni, che si riferiscono ai componenti eseguiti all'interno di un cluster Kubernetes, questa guida tratta i seguenti argomenti:

- Distribuzione dei carichi di lavoro tra nodi e zone di disponibilità
- Protezione dei carichi di lavoro critici con un PDB
- Configurazione di sonde e controlli sanitari
- Configurazione degli hook del ciclo di vita dei container
- Comprensione dello sfratto dei pod durante le interruzioni zonali

Distribuisci i carichi di lavoro tra nodi e zone di disponibilità

La distribuzione di un carico di lavoro tra domini di errore come zone di disponibilità e nodi migliora la disponibilità dei componenti e riduce le possibilità di errore per le applicazioni con scalabilità

orizzontale. Nelle sezioni seguenti vengono illustrati i modi per distribuire i carichi di lavoro tra nodi e zone di disponibilità.

Utilizza i vincoli di diffusione della topologia dei pod

I vincoli di diffusione della topologia dei pod Kubernetes indicano allo scheduler Kubernetes di distribuire i pod gestiti da `ReplicaSet` o `StatefulSet` tra diversi domini di errore (zone di disponibilità, nodi e tipi di hardware). Quando utilizzi i vincoli di diffusione della topologia dei pod, puoi fare quanto segue:

- Distribuisci o concentra i pod su diversi domini di errore a seconda dei requisiti dell'applicazione. Ad esempio, è possibile distribuire i pod per garantire la resilienza e concentrare i pod per le prestazioni di rete.
- Combina diverse condizioni, ad esempio la distribuzione tra zone di disponibilità e la distribuzione tra nodi.
- Specificate l'azione preferita se le condizioni non possono essere soddisfatte:
  - `whenUnsatisfiable: DoNotSchedule`Utilizzalo con una combinazione di `maxSkew` e `minDomains` per creare requisiti rigidi per lo scheduler.
  - Usa `whenUnsatisfiable: ScheduleAnyway` per ridurre`maxSkew`.

Se una zona guasta non è più disponibile, i pod in quella zona diventano malsani. Kubernetes riprogramma i pod rispettando, se possibile, il vincolo di diffusione.

Il codice seguente mostra un esempio di utilizzo dei vincoli di diffusione della topologia dei pod tra le zone di disponibilità o tra i nodi:

```
...
spec:
  selector:
    matchLabels:
      app: <your-app-label>
    replicas: 3
    template:
      metadata:
        labels: <your-app-label>
      spec:
        serviceAccountName: <ServiceAccountName>
...
```

```
      topologySpreadConstraints:
      - labelSelector:
          matchLabels:
            app: <your-app-label>
        maxSkew: 1
        topologyKey: topology.kubernetes.io/zone # <---spread those pods evenly over
all availability zones
        whenUnsatisfiable: ScheduleAnyway
      - labelSelector:
          matchLabels:
            app: <your-app-label>
        maxSkew: 1
        topologyKey: kubernetes.io/hostname # <---spread those pods evenly over all
nodes
        whenUnsatisfiable: ScheduleAnyway
```

Vincoli di diffusione della topologia predefiniti a livello di cluster

Per impostazione predefinita, Kubernetes fornisce una serie di vincoli di diffusione della topologia per la distribuzione dei pod tra nodi e zone di disponibilità:

```
defaultConstraints:
  - maxSkew: 3
    topologyKey: "kubernetes.io/hostname"
    whenUnsatisfiable: ScheduleAnyway
  - maxSkew: 5
    topologyKey: "topology.kubernetes.io/zone"
    whenUnsatisfiable: ScheduleAnyway
```

> **Note**
>
> Le applicazioni che richiedono diversi tipi di vincoli di topologia possono sovrascrivere la policy a livello di cluster.

I vincoli predefiniti impostano un valore massimo`maxSkew`, il che non è utile per le implementazioni con un numero limitato di pod. Al momento, non `KubeSchedulerConfiguration` può essere modificato in Amazon EKS. Se devi applicare altri set di vincoli di diffusione della topologia, prendi in considerazione l'utilizzo di un controller di ammissione mutante come nella sezione seguente. Puoi anche controllare i vincoli di diffusione della topologia predefiniti se esegui uno scheduler alternativo.

Tuttavia, la gestione di pianificatori personalizzati aggiunge complessità e può avere implicazioni sulla resilienza dei cluster e sull'HA. Per questi motivi, non è consigliabile utilizzare uno scheduler alternativo solo per i vincoli di diffusione della topologia.

La politica di Gatekeeper per i vincoli di diffusione della topologia

Un'altra opzione per applicare i vincoli di diffusione della topologia consiste nell'utilizzare una politica del progetto Gatekeeper. Le policy di Gatekeeper sono definite a livello di applicazione.

I seguenti esempi di codice mostrano l'uso di una `Gatekeeper OPA` policy per la distribuzione. È possibile modificare la politica in base alle proprie esigenze. Ad esempio, applica la policy solo alle distribuzioni che hanno l'etichetta o `HA=true` scrivi una policy simile utilizzando un policy controller diverso.

Questo primo esempio mostra l'`ConstraintTemplate`utilizzo con: `k8stopologyspreadrequired_template.yml`

```
apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8stopologyspreadrequired
spec:
  crd:
    spec:
      names:
        kind: K8sTopologySpreadRequired
      validation:
        openAPIV3Schema:
          type: object
          properties:
            message:
              type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8stopologyspreadrequired

        get_message(parameters, _default) =3D msg {
          not parameters.message
          msg :=_default
        }
```

```
        get_message(parameters, _default) =3D msg {
          msg := parameters.message
        }

        violation[{"msg": msg}] {
          input.review.kind.kind ="Deployment"
          not input.review.object.spec.template.spec.topologySpreadConstraint
          def_msg :"Pod Topology Spread Constraints are required for Deployments"
          msg :get_message(input.parameters, def_msg)

        }
```

Il codice seguente mostra il manifesto `constraints` YAML: `k8stopologyspreadrequired_constraint.yml`

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sTopologySpreadRequired
metadata:
  name: require-topologyspread-for-deployments
spec:
  match:
    kinds:
      - apiGroups: ["apps"]
        kinds: ["Deployment"]
    namespaces:  ## Without theses two lines will apply to the whole cluster
      - "example"
```

Quando utilizzare i vincoli di diffusione della topologia

Prendi in considerazione l'utilizzo dei vincoli di diffusione della topologia per i seguenti scenari:

- Qualsiasi applicazione scalabile orizzontalmente (ad esempio, servizi web stateless)
- Applicazioni con repliche attive-attive o attive-passive (ad esempio database o cache NoSQL)
- Applicazioni con repliche in standby (ad esempio controller)

I componenti di sistema che possono essere utilizzati per lo scenario con scalabilità orizzontale, ad esempio, includono quanto segue:

- Cluster Autoscaler e Karpenter (con e) `replicaCount > 1` `leader-elect = true`

- AWS Controller Load Balancer
- CoreDNS

Affinità e antiaffinità dei pod

In alcuni casi, è utile assicurarsi che non più di un pod di un tipo specifico sia in esecuzione su un nodo. Ad esempio, per evitare di programmare più pod che utilizzano molte risorse di rete sullo stesso nodo, puoi utilizzare la regola di antiaffinità con l'etichetta o. `Ingress` `Network-heavy` Quando si utilizza`anti-affinity`, è anche possibile utilizzare una combinazione dei seguenti elementi:

- Influenze sui nodi ottimizzati per la rete
- Tolleranze corrispondenti su pod con un elevato numero di connessioni di rete
- Affinità dei nodi o selettore di nodi per garantire che i pod che utilizzano molte risorse di rete utilizzino istanze ottimizzate per la rete

Network-heavy i pod vengono utilizzati come esempio. Potresti avere requisiti diversi, come GPU, memoria o archiviazione locale. Per altri esempi di utilizzo e opzioni di configurazione, consulta la documentazione di Kubernetes.

I pod Rebalance

Questa sezione illustra due approcci al ribilanciamento dei pod in un cluster Kubernetes. Il primo utilizza il Descheduler per Kubernetes. Il Descheduler aiuta a mantenere la distribuzione dei pod applicando strategie per rimuovere i pod che violano i vincoli di diffusione della topologia o le regole di antiaffinità. Il secondo approccio utilizza la funzionalità di consolidamento e bin-packing di Karpenter. Il consolidamento valuta e ottimizza continuamente l'utilizzo delle risorse consolidando i carichi di lavoro su un numero inferiore di nodi e in modo più efficiente.

Ti consigliamo di utilizzare Descheduler se non utilizzi Karpenter. Se utilizzi Karpenter e Cluster Autoscaler insieme, puoi usare Descheduler con Cluster Autoscaler per i gruppi di nodi.

Descheduler per nodi senza gruppi

Non c'è alcuna garanzia che i vincoli di topologia rimangano soddisfatti quando i pod vengono rimossi. Ad esempio, il ridimensionamento di un'implementazione potrebbe comportare uno squilibrio

nella distribuzione dei pod. Tuttavia, poiché Kubernetes utilizza i vincoli di diffusione della topologia dei pod solo nella fase di pianificazione, i pod rimangono sbilanciati nel dominio di errore.

Per mantenere una distribuzione bilanciata dei pod in tali scenari, puoi utilizzare Descheduler for Kubernetes. Descheduler è uno strumento utile per molteplici scopi, ad esempio per imporre l'età massima dei pod o il time to live (TTL) o per migliorare l'uso dell'infrastruttura. Nel contesto della resilienza e dell'alta disponibilità (HA), considera le seguenti strategie di Descheduler:

- RemovePodsViolatingTopologySpreadConstraint
- RemovePodsViolatingInterPodAntiAffinity
- RemoveDuplicates

Funzionalità di consolidamento e bin-packing di Karpenter

Per i carichi di lavoro che utilizzano Karpenter, puoi utilizzare la funzionalità di consolidamento e bin-packing per ottimizzare l'utilizzo delle risorse e ridurre i costi nei cluster Kubernetes. Karpenter valuta continuamente il posizionamento dei pod e l'utilizzo dei nodi e tenta di consolidare i carichi di lavoro su un numero inferiore di nodi e, ove possibile, impacchettati in modo più efficiente. Questo processo prevede l'analisi dei requisiti di risorse, la considerazione di vincoli come le regole di affinità dei pod e il potenziale spostamento dei pod tra i nodi per migliorare l'efficienza complessiva del cluster. Il codice seguente fornisce un esempio:

```
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: default
spec:
  disruption:
    consolidationPolicy: WhenUnderutilized
    expireAfter: 720h
```

Per`consolidationPolicy`, puoi usare `WhenUnderutilized` o`WhenEmpty`:

- Quando `consolidationPolicy` è impostato su`WhenUnderutilized`, Karpenter considera tutti i nodi per il consolidamento. Quando Karpenter scopre un nodo vuoto o sottoutilizzato, Karpenter tenta di rimuovere o sostituire il nodo per ridurre i costi.
- Quando `consolidationPolicy` è impostato su`WhenEmpty`, Karpenter considera per il consolidamento solo i nodi che non contengono pod per il carico di lavoro.

Le decisioni di consolidamento di Karpenter non si basano esclusivamente sulle percentuali di utilizzo della CPU o della memoria che potresti riscontrare negli strumenti di monitoraggio. Karpenter utilizza invece un algoritmo più complesso basato sulle richieste di risorse dei pod e sulle potenziali ottimizzazioni dei costi. Per ulteriori informazioni, consulta la documentazione di Karpenter.

Proteggi i carichi di lavoro critici con un PDB

Un pod disruption budget (PDB) è una funzionalità essenziale per mantenere l'elevata disponibilità delle applicazioni in un cluster. Il PDB specifica una dimensione di destinazione, che è la disponibilità minima per un particolare tipo di pod. Ciò significa che un numero minimo di repliche di un particolare tipo di pod deve essere in esecuzione in un dato momento. Se il numero di repliche in esecuzione scende al di sotto della dimensione prevista, Kubernetes impedisce ulteriori interruzioni delle repliche rimanenti fino al raggiungimento della dimensione prevista. I PDB aiutano a garantire che i carichi di lavoro non siano influenzati da questi eventi e possano continuare a funzionare senza interruzioni. Quando si verifica un'interruzione, Kubernetes tenta di rimuovere correttamente i pod dai nodi interessati mantenendo il numero di repliche specificato nel PDB.

È possibile utilizzare un PDB per dichiarare il numero e il numero di repliche. `minAvailable` `maxUnavailable` Ad esempio, se desideri che siano disponibili almeno tre copie della tua app, crea un PDB simile all'esempio seguente:

```
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: my-svc-pdb
spec:
  minAvailable: 3
  selector:
    matchLabels:
      app: my-svc
```

La corretta configurazione dei PDB per le applicazioni consente di ridurre al minimo le interruzioni durante eventi pianificati o non pianificati. È possibile utilizzare la regola anti-affinità per pianificare i pod di una distribuzione su nodi diversi ed evitare ritardi PDB durante gli aggiornamenti dei nodi.

Configura le sonde e i controlli dello stato del load balancer

Kubernetes offre diversi modi per eseguire i controlli dello stato delle applicazioni oltre ai controlli dello stato del bilanciamento del carico. Puoi eseguire le seguenti sonde integrate in Kubernetes insieme al controllo dello stato del bilanciamento del carico come comando nel contesto del pod o come sonda verso HTTP/TCP kubelet o l'indirizzo IP dell'host.

Le sonde Liveness e le sonde Readiness devono essere diverse e indipendenti (o almeno con valori di timeout diversi). Se un'applicazione presenta un problema temporaneo, la sonda di prontezza contrassegnerà il pod come non pronto fino a quando il problema non sarà risolto. Se le impostazioni della sonda di vivacità non sono corrette, la sonda di vivacità potrebbe far terminare il pod.

Sonda di avvio

Utilizzate le sonde di avvio per proteggere le applicazioni con cicli di inizializzazione lunghi. Fino a quando la sonda di avvio non ha esito positivo, le altre sonde sono disattivate.

Puoi definire un tempo massimo di attesa per Kubernetes per l'avvio dell'applicazione. Se, dopo il tempo massimo configurato, il pod continua a non rispondere ai test di avvio, l'applicazione viene terminata e viene creato un nuovo pod.

Utilizzate le sonde di avvio quando l'ora di avvio di un'applicazione è imprevedibile. Se sapete che l'applicazione richiede 10 secondi per avviarsi, utilizzate invece una sonda di vivacità o una sonda di prontezza. `initialDelaySeconds`

Sonda di vivacità

Utilizza le sonde Liveness per rilevare i problemi delle applicazioni o se il processo è in esecuzione senza problemi. Una sonda di liveness è in grado di rilevare condizioni di stallo in cui il processo continua a funzionare ma l'applicazione non risponde. Quando utilizzate una sonda di vivacità, effettuate le seguenti operazioni:

- `initialDelaySeconds`Usare per ritardare la prima sonda.
- Non impostate le stesse specifiche per le sonde di vivacità e prontezza.
- Non configurate una sonda di liveness in modo che dipenda da un fattore esterno al vostro pod (ad esempio, un database).
- Imposta la sonda di vivacità per uno specifico. `terminationGracePeriodSeconds` Per ulteriori informazioni, consulta la documentazione di Kubernetes.

Sonda di prontezza

Utilizzate una sonda di prontezza per rilevare quanto segue:

- Se l'applicazione è pronta ad accettare il traffico
- Disponibilità parziale, in cui l'applicazione potrebbe essere temporaneamente non disponibile ma dovrebbe essere nuovamente integra dopo il completamento di una determinata operazione

Le sonde di prontezza aiutano a garantire che la configurazione e le dipendenze dell'applicazione vengano eseguite senza problemi o errori, in modo che l'applicazione possa servire il traffico. Tuttavia, una sonda di prontezza configurata in modo errato può causare un'interruzione anziché prevenirla. Le sonde di prontezza che dipendono da fattori esterni, come la connettività a un database, possono causare il malfunzionamento della sonda in tutti i pod. Tali errori possono causare un'interruzione e un guasto a catena da un servizio di backend ad altri servizi che utilizzavano i pod guasti.

Controlli dello stato delle risorse e del sistema di bilanciamento del carico in ingresso

Application Load Balancer e `ingress` Kubernetes offrono funzionalità di controllo dello stato. Per i controlli di integrità dell'Application Load Balancer, specifica le porte e il percorso di destinazione.

> **Note**
>
> Per Kubernetes`ingress`, ci sarà una latenza di annullamento della registrazione. L'impostazione predefinita per Application Load Balancer è 300 secondi. Valuta la possibilità di configurare il controllo dello stato delle risorse in ingresso o del load balancer utilizzando gli stessi valori utilizzati per la sonda di prontezza.

NGINX fornisce anche un controllo dello stato di salute. Per ulteriori informazioni, consulta la documentazione di NGINX.

I gateway di ingresso e uscita Istio non dispongono di un meccanismo di controllo dello stato paragonabile al controllo dello stato di integrità HTTP di NGINX. Tuttavia, è possibile ottenere funzionalità simili utilizzando l'interruttore automatico Istio o il rilevamento dei valori anomali. `DestinationRule`

Per ulteriori informazioni, consulta Availability and Pod Lifecycle nella Amazon EKS Best Practices Guide.

Configura i ganci per il ciclo di vita dei container

Durante uno spegnimento regolare del contenitore, l'applicazione dovrebbe rispondere a un `SIGTERM` segnale avviandone lo spegnimento, in modo che i client non subiscano tempi di inattività. L'applicazione deve eseguire procedure di pulizia come le seguenti:

- Salvataggio dei dati
- Chiusura dei descrittori di file
- Chiusura delle connessioni al database
- Completamento delle richieste in volo con garbo
- Uscire tempestivamente per soddisfare la richiesta di terminazione del pod

Imposta un periodo di grazia sufficientemente lungo per completare la pulizia. Per informazioni su come rispondere al `SIGTERM` segnale, consultate la documentazione relativa al linguaggio di programmazione utilizzato per l'applicazione.

Gli hook del ciclo di vita dei container consentono ai contenitori di essere consapevoli degli eventi del loro ciclo di vita di gestione. I contenitori possono eseguire il codice implementato in un gestore quando viene eseguito l'hook del ciclo di vita corrispondente. Gli hook del ciclo di vita dei container forniscono una soluzione alternativa per la natura asincrona di Kubernetes e del cloud. Questo approccio può prevenire la perdita delle connessioni che vengono inoltrate al pod di terminazione prima che la risorsa di ingresso entri e vengono aggiornate per non inviare nuovo traffico al pod. `iptables`

Ciclo di vita dei container e `EndpointSlice` fanno parte di `Endpoint` diverse API. È importante orchestrare queste API. Tuttavia, quando un pod viene chiuso, l'API Kubernetes notifica contemporaneamente sia il kubelet (per il ciclo di vita del contenitore) che il controller. `EndpointSlice` Per ulteriori informazioni, incluso un diagramma, consulta Gestire con garbo le richieste dei clienti nella Amazon EKS Best Practices Guide.

Quando viene `kubelet` inviato `SIGTERM` al pod, il `EndpointSlice` controller termina l'oggetto. `EndpointSlice` Tale terminazione notifica ai server dell'API Kubernetes di notificare ogni nodo da aggiornare. `kube-proxy` `iptables` Sebbene queste azioni avvengano contemporaneamente, non

vi sono dipendenze o sequenze tra di esse. È molto probabile che il contenitore riceva il `SIGKILL` segnale molto prima dell'aggiornamento delle regole locali `kube-proxy` `iptables` su ciascun nodo. In tal caso, gli scenari possibili includono quanto segue:

- Se la tua richiesta elimina immediatamente e senza mezzi termini le richieste e le coincidenze in volo al momento della ricezione`SIGTERM`, i clienti riscontreranno degli errori. `500`
- Se l'applicazione garantisce che tutte le richieste e le coincidenze in volo vengano elaborate completamente al momento della ricezione`SIGTERM`, durante il periodo di prova, le nuove richieste dei clienti verranno comunque inviate al contenitore dell'applicazione perché `iptables` le regole potrebbero non essere ancora aggiornate. Fino a quando la procedura di pulizia non chiuderà il socket del server sul contenitore, tali nuove richieste comporteranno nuove connessioni. Al termine del periodo di prova, le nuove connessioni stabilite dopo l'`SIGTERM`invio vengono interrotte incondizionatamente.

Per risolvere gli scenari precedenti, puoi implementare l'integrazione in-app o il PreStop lifecycle hook. Per ulteriori informazioni, incluso un diagramma, consulta Gracefully shutdown applications nella Amazon EKS Best Practices Guide.

> **Note**
>
> Indipendentemente dal fatto che l'applicazione si chiuda correttamente o che sia il risultato dell'`preStop`hook, i contenitori dell'applicazione vengono terminati alla fine del periodo di prova. `SIGKILL`

Utilizzate l'`preStop`hook con un `sleep` comando per ritardare l'invio. `SIGTERM` Ciò contribuirà a continuare ad accettare le nuove connessioni mentre l'oggetto di ingresso le indirizza al pod. Verifica il valore temporale del `sleep` comando per assicurarti che venga presa in considerazione l'eventuale latenza di Kubernetes e altre dipendenze dell'applicazione, come mostrato nell'esempio seguente:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      lifecycle:
```

```
        # This "sleep" preStop hook delays the Pod shutdown until
        # after the Ingress Controller removes the matching Endpoint or EndpointSlice
        preStop:
          exec:
            command:
              - /bin/sleep
              - "20"
              # This period should be turned to Ingress/Service Mesh update latency
```

Per ulteriori informazioni, consulta Container hooks nella documentazione di Kubernetes e Gracefully shutdown delle applicazioni nella Amazon EKS Best Practices Guide.

Comprendi lo sfratto dei pod durante le interruzioni zonali

Quando si verifica un'interruzione completa della zona di disponibilità, ovvero quando tutti i nodi di quella zona di disponibilità perdono la connettività al piano di controllo di Kubernetes, il controller del ciclo di vita del nodo in Kubernetes rileva la situazione ed elimina i pod dalla zona interessata. I pod sui nodi non raggiungibili sono contrassegnati come e i nuovi pod sono programmati su nodi integri nelle zone di disponibilità disponibili. `Terminating` Durante questo periodo, i nodi interessati mostrano uno `NotReady` stato, lo scheduler impedisce l'inserimento di nuovi pod su tali nodi e il EndpointSlice controller rimuove gli endpoint associati alla zona di disponibilità compromessa dal routing del servizio fino al ripristino della connettività.

Per gli scenari che comportano guasti parziali dei nodi all'interno di una zona, in cui solo un sottoinsieme di nodi diventa irraggiungibile, il controller del ciclo di vita del nodo applica un comportamento di sfratto diverso. Se l'interruzione persiste oltre il periodo di tolleranza configurato (per impostazione predefinita, cinque minuti), i pod sui nodi disconnessi vengono contrassegnati come e i nuovi pod vengono programmati su nodi integri nelle zone di disponibilità disponibili. `Terminating`

Implementazione del trasferimento zonale di Amazon EKS per una migliore resilienza

Il trasferimento di zona di Amazon EKS, che si integra con Amazon Application Recovery Controller (ARC), fornisce un meccanismo per gestire in modo proattivo il traffico durante le limitazioni della zona di disponibilità. Questa funzionalità consente il reindirizzamento temporaneo del traffico di rete da una zona di disponibilità non integra verso zone integre all'interno della stessa per ridurre al minimo le interruzioni del servizio. Regione AWS

Comprensione del meccanismo di spostamento zonale

Lo spostamento di zona di Amazon EKS affronta il traffico est-ovest (comunicazione tra pod all'interno del cluster). Quando lo spostamento zonale è configurato con Application Load Balancers o Network Load Balancers, supporta anche il routing del traffico in ingresso. Il meccanismo funziona coordinando più componenti di Kubernetes e del piano di AWS controllo per reindirizzare il traffico in modo sicuro senza interrompere i carichi di lavoro in esecuzione. Durante un cambio di zona attivo, Amazon EKS esegue automaticamente le seguenti azioni coordinate:

- Cordonamento dei nodi: tutti i nodi nella zona di disponibilità ridotta sono cordonati. Ciò impedisce allo scheduler Kubernetes di inserire nuovi pod sui nodi mentre mantiene i carichi di lavoro esistenti.
- Sospensione del ribilanciamento delle zone di disponibilità: per i gruppi di nodi gestiti, le operazioni di ribilanciamento delle zone di disponibilità vengono sospese e i gruppi di Auto Scaling vengono aggiornati per lanciare nuovi nodi del piano dati esclusivamente nelle zone di disponibilità integre. Ciò garantisce che non venga fornita nuova capacità nella zona compromessa.
- Rimozione degli endpoint: il EndpointSlice controller rimuove gli endpoint dei pod nella zona di disponibilità compromessa da tutti i dispositivi pertinenti. EndpointSlices Ciò garantisce che i meccanismi di rilevamento dei servizi e di bilanciamento del carico indirizzino il traffico solo verso i pod che funzionano in zone di disponibilità funzionanti.
- Conservazione del carico di lavoro: Amazon EKS si astiene dal chiudere i nodi o rimuovere i pod nella zona di disponibilità interessata. Mantiene la piena capacità nella zona compromessa in modo che, quando il turno zonale scade o viene annullato, il traffico possa rientrare in sicurezza senza richiedere ulteriori operazioni di ridimensionamento.

Metodi di attivazione del turno zonale

È possibile scegliere tra due approcci per avviare i cambiamenti zonali, a seconda del modello operativo:

- Lo spostamento zonale manuale fornisce il controllo guidato dall'operatore quando vengono rilevati problemi specifici relativi alla zona di disponibilità tramite monitoraggio, avvisi o segnalazioni dei clienti. Questo metodo richiede un'azione esplicita tramite la console ARC, AWS Command Line Interface (AWS CLI) o le API zonal shift, in cui gli operatori specificano la zona di disponibilità compromessa e definiscono un orario di scadenza per il turno. I turni manuali sono appropriati

quando i team dispongono di funzionalità dedicate di monitoraggio e di chiamata e preferiscono mantenere il controllo diretto sulle decisioni di gestione del traffico.

- L'autoshift zonale AWS autorizza ad avviare automaticamente i turni quando ARC rileva potenziali guasti o alterazioni della zona di disponibilità sulla base di segnali telemetrici e sanitari interni su più segnali, tra cui metriche di rete Servizi AWS, Amazon Elastic Compute Cloud (Amazon EC2) ed Elastic Load Balancing. AWS termina automaticamente un cambio automatico quando gli indicatori indicano che il problema è stato risolto. Se si desidera garantire la massima disponibilità con un intervento manuale minimo, si consiglia questo approccio, in quanto consente di rispondere in meno di un minuto ai problemi rilevati nella zona di disponibilità.

Prerequisiti per un effettivo cambiamento di zona

Affinché lo spostamento zonale protegga con successo le applicazioni in caso di problemi relativi alla zona di disponibilità, è necessario progettare i cluster in modo da garantire la Multi-AZ resilienza prima di abilitare la funzionalità di spostamento zonale:

- Multi-AZ distribuzione dei nodi: esegui il provisioning dei nodi di lavoro in almeno tre zone di disponibilità per garantire una ridondanza sufficiente quando una zona diventa non disponibile.
- Pianificazione della capacità: capacità di elaborazione Pre-provision sufficiente nelle zone di disponibilità integre per soddisfare l'intero carico di lavoro quando una zona di disponibilità viene rimossa dal servizio, poiché le operazioni di scalabilità durante un'interruzione attiva potrebbero comportare una capacità insufficiente.
- Distribuzione su pod e pre-scalabilità: distribuisci più repliche di ogni applicazione in tutte le zone di disponibilità e prescala componenti di sistema critici come CoredNS in ogni zona. Questo aiuta a garantire che rimanga una capacità sufficiente dopo lo spostamento di una zona.

Raccomandazioni per la resilienza alle perturbazioni zonali

- Abilita lo spostamento di zona alla creazione del cluster: per i nuovi cluster EKS, abilita l'integrazione dello spostamento di zona con ARC durante il provisioning iniziale tramite la console Amazon EKS o strumenti Infrastructure as Code (IaC) come. AWS CLI AWS CloudFormation I cluster EKS Auto Mode creati con configurazione rapida hanno lo spostamento zonale abilitato per impostazione predefinita.
- Seleziona il metodo di attivazione appropriato: scegli lo spostamento automatico zonale per ambienti di produzione che richiedono la massima disponibilità con risposta automatizzata, in

particolare per le applicazioni rivolte ai clienti in cui minuti di inattività durante una violazione della zona di disponibilità possono avere un impatto aziendale significativo. Utilizza lo spostamento zonale manuale per ambienti in cui i team operativi preferiscono fornire un'approvazione esplicita prima dei cambiamenti di traffico o in cui i test e la convalida delle applicazioni sono ancora in corso.

- Verifica la resilienza prima dell'implementazione in produzione: convalida il comportamento dei cluster in caso di Single-AZ perdita avviando manualmente i turni zonali di test o abilitando le procedure di trasferimento automatico delle zone per verificare che le applicazioni mantengano la disponibilità, le prestazioni rimangano accettabili e la capacità sia sufficiente quando si opera con un numero ridotto di zone di disponibilità. Consigliamo vivamente questo test in modo da poter identificare le lacune di configurazione prima che si verifichino danni effettivi alla zona di disponibilità.
- Coordina la configurazione con il load balancer: per le applicazioni che ricevono traffico esterno, abilita lo spostamento zonale ARC sugli Application Load Balancer e Network Load Balancer associati per garantire che il traffico in ingresso e il traffico est-ovest all'interno del cluster si spostino contemporaneamente durante i problemi relativi alla zona di disponibilità. Questo coordinamento previene gli scenari in cui le richieste esterne raggiungano i pod integri ma tali pod non possono comunicare con le dipendenze nella zona spostata.
- Monitora le operazioni a turni: dopo aver abilitato il cambiamento zonale, configura il monitoraggio e gli avvisi per gli eventi relativi ai turni, tra cui le attivazioni dello spostamento automatico, l'avvio manuale dei turni e le scadenze dei turni, per mantenere la visibilità operativa sulle azioni di gestione del traffico e sul loro impatto sul comportamento delle applicazioni.

Completamento e ripristino dei turni

Quando un turno di zona scade in base alla durata configurata o viene annullato manualmente dopo la risoluzione del problema relativo alla zona di disponibilità, il EndpointSlice controller aggiorna automaticamente tutto EndpointSlices per reincorporare gli endpoint nella zona di disponibilità ripristinata. Il traffico torna gradualmente alla zona interessata in precedenza man mano che i client aggiornano le informazioni sugli endpoint e stabiliscono nuove connessioni. Ciò consente l'utilizzo completo della capacità del cluster senza richiedere l'intervento manuale o la riprogrammazione dei pod.

Conclusioni

Quando progetti la tua architettura per un'elevata disponibilità e resilienza delle applicazioni, considera i seguenti componenti:

- L'applicazione per i microservizi (i relativi pod e contenitori)
- Il piano dati del carico di lavoro (Ingress Controller, pod, componenti di sistema come Amazon VPC CNI, service mesh sidecar e kube-proxy)
- Il livello di gestione del carico di lavoro (controller, controller di ammissione, motori di policy di rete e archiviazione persistente dei dati per questi componenti)
- Il piano di controllo di Kubernetes
- Infrastruttura (nodi, rete e dispositivi di rete)

Per rispondere a tali considerazioni sui componenti, utilizza le seguenti strategie chiave:

- Per garantire un'elevata disponibilità e tolleranza agli errori, distribuisci i carichi di lavoro tra nodi e zone di disponibilità.
- Per proteggere i carichi di lavoro critici, mantieni la stabilità delle applicazioni durante le interruzioni utilizzando pod disruption budgets (). PDBs
- Per garantire che i pod funzionino e servano il traffico correttamente, configurate le sonde di avvio, le sonde di liveness, le sonde di prontezza e i controlli dello stato del sistema di bilanciamento del carico.
- Per gestire in modo efficiente le transizioni di stato dei container, configura gli hook del ciclo di vita dei container.
- Per controllare il processo di sfratto durante i guasti o la manutenzione dei nodi, configura il tempo di sfratto del pod.

Implementando queste pratiche, puoi migliorare in modo significativo l'affidabilità e la resilienza delle applicazioni in esecuzione su Amazon EKS, garantendo prestazioni solide e alta disponibilità.

Resources

- Vincoli di diffusione della topologia Kubernetes Pod (documentazione Kubernetes)
- FAQsKarpenter (documentazione Karpenter)
- Descheduler per Kubernetes (repository) GitHub
- Disponibilità e ciclo di vita dei pod: Guida alle best practice di Amazon EKS
- Chiudi correttamente le applicazioni Amazon EKS Best Practices Guide
- [EKS] [richiesta]: Capacità di configurazione pod-eviction-timeout e soluzioni alternative (repository Containers Roadmap)

Cronologia dei documenti

La tabella seguente descrive le modifiche significative apportate a questa guida. Per ricevere notifiche sugli aggiornamenti futuri, puoi abbonarti a un feed RSS.

Modifica	Descrizione	Data
Aggiorna	È stata rivista la sezione sugli sfratti dei pod durante le interruzioni zonali.	29 ottobre 2025
Aggiorna	È stata rivista la sezione Usa i vincoli di diffusione della topologia dei pod.	27 gennaio 2025
Pubblicazione iniziale	—	23 ottobre 2024

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.