STRATEGIE DI RILASCIO CON AMAZON ECS: IL BLUE/GREEN DEPLOYMENT.

Amazon ECS | blue/green deployment | CI/CD | Continuous Integration

beSharp | 16 Ottobre 2020

L'**architettura a microservizi** è diventata ormai predominante in molti progetti software. Grazie ai suoi enormi vantaggi in termini di velocità di sviluppo e di deploy, agevola i task quotidiani dei DevOps e non solo.

Per sfruttare questo pattern architetturale su AWS possiamo utilizzare **Amazon Elastic Container Service (Amazon ECS)**, un servizio di orchestrazione dei container completamente gestito.

ECS è una grande scelta per l'esecuzione dei container per diverse ragioni. Per prima cosa, permette di scegliere di eseguire i cluster ECS utilizzando AWS Fargate, un'infrastruttura serverless per container. Fargate rimuove la necessità di effettuare il provisioning e la gestione dei server, permettendoci di pagare solo per le risorse utilizzate per ciascuna applicazione.

Inoltre, ECS può essere integrato in modo nativo con altri servizi come Amazon Route 53, Secrets Manager, AWS Identity and Access Management (IAM) e Amazon CloudWatch.

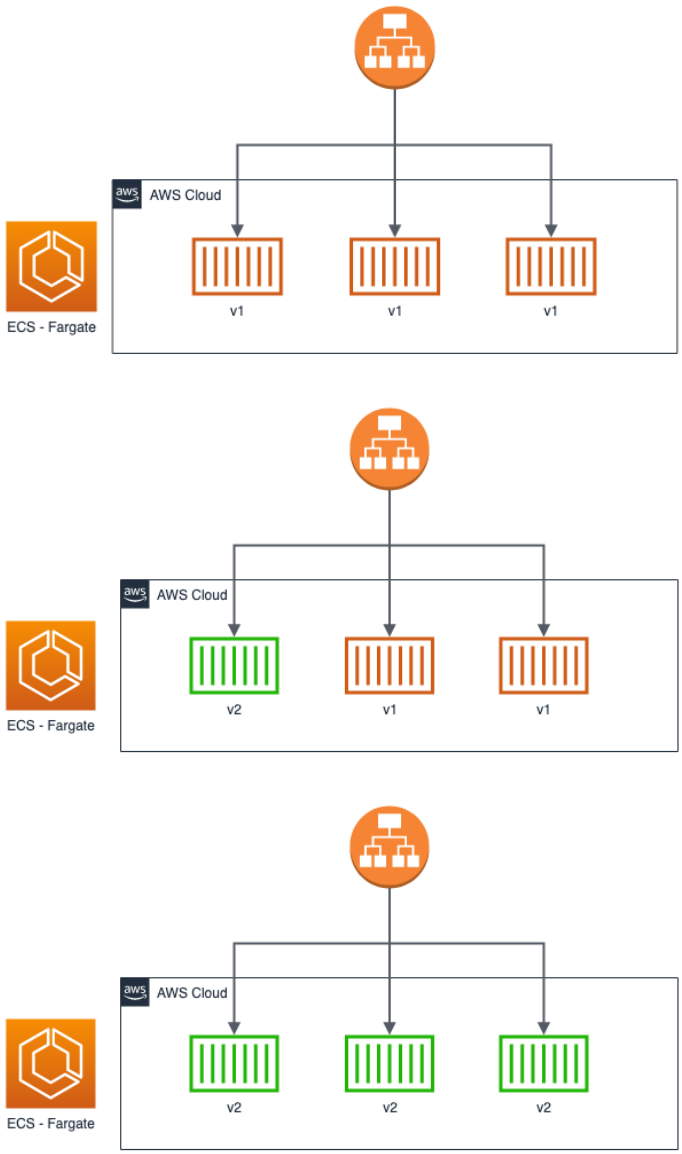
Gestire architetture a microservizi su ECS non è sempre banale. Sul nostro blog abbiamo già parlato di come creare e gestire cluster e implementare pipeline di CI/CD per ECS.

In questo articolo andremo invece a vedere un aspetto fondamentale per la **continuous integration** del software, ovvero le diverse modalità di rilascio dei nuovi pacchetti software. In particolare, ci focalizzeremo sulla metodologia **blue/green**, una tecnica relativamente nuova che risolve molti dei problemi presenti con altre modalità di deployment più semplici.

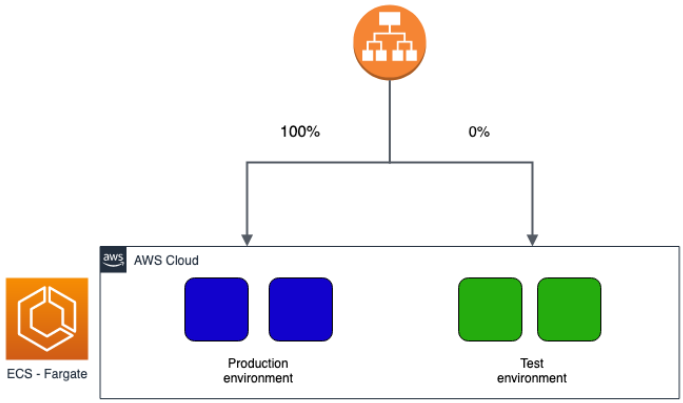
Strategie di deployment

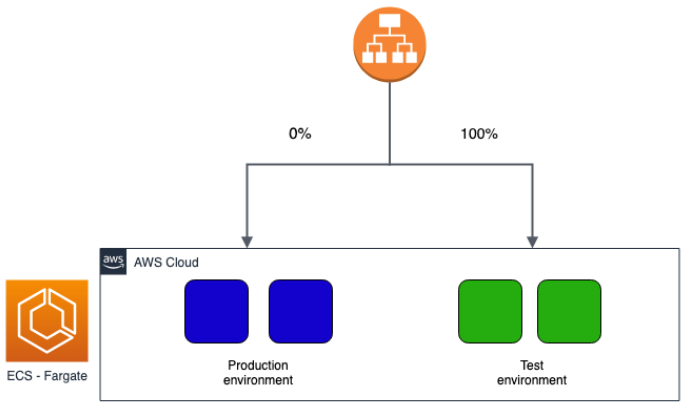
Ad oggi esistono numerose strategie di deployment. Vediamo insieme quelle più diffuse e supportate da ECS:

- **Rolling update**: Con questa modalità, la vecchia e la nuova versione software coesisteranno nella fase di rilascio. Infatti, il nuovo pacchetto verrà installato progressivamente su tutti i server. Cosa comporta? La tua applicazione dev'essere in grado di supportare entrambe le versioni, il che può diventare complicato quando queste due utilizzano una versione diversa del database. Questa modalità di deployment è tra le più semplici da setuppare, tuttavia è importante considerare che prevede forti vincoli sulla retrocompatibilità delle versioni rilasciate.



- **Blue/green**: A differenza del caso precedente, qui è possibile mantenere la vecchia infrastruttura (*blue*) nel mentre che la nuova versione software viene installata in una nuova infrastruttura temporanea (*green*). Una volta che questa è installata, è possibile effettuare integration/validation test sulla nuova infrastruttura per capire se possiamo effettivamente promuovere il nuovo pacchetto software. Se così fosse, lo switch del traffico può essere svolto praticamente senza downtime. La vecchia infrastruttura può essere mantenuta per un certo lasso di tempo per permettere eventuali operazioni di rollback, infine può essere decommissionata.





- **Canary**: Questa tipologia è molto simile alla precedente, ma a differenza di essa prevede un graduale dirottamento di traffico dalla vecchia alla nuova infrastruttura (lineare o a step configurabili)

Ovviamente sta a noi decidere quale tipologia scegliere in base alle nostre esigenze. Sfruttando i vantaggi del Cloud può convenire prediligere una tipologia blue/green o canary rispetto ad una rolling update per evitare downtime e avere rapidi time di rollback nel caso ce ne fosse la necessità. Ad oggi, implementare queste nuove tipologie di deployment è relativamente semplice e a costo bassissimo, soprattutto se si usano tecnologie serverless come ECS in modalità Fargate!

Prerequisiti

Lo scopo di questo articolo è **analizzare step-by-step come realizzare una gestione di blue/green deployment su ECS**. Prima di iniziare, però, è necessario soddisfare i alcuni requisiti:

- Avere un cluster ECS in modalità Fargate: Abbiamo a nostra disposizione numerosi strumenti: la CLI di AWS, le API o la Console. Per i più “avventurosi” è possibile anche utilizzare la ECS CLI come abbiamo spiegato in questo recente articolo.

  Altrimenti, se volete utilizzare l’approccio utilizzato in questo articolo, potete creare il vostro cluster ECS in modalità Fargate da console. Ecco come farlo.

- Avere un Application Load Balancer (AWS ELB) configurato con 2 listener associati a 2 target group, come riportato di seguito. Lo utilizzeremo più tardi nell’articolo durante la Configurazione ECS – Fargate.

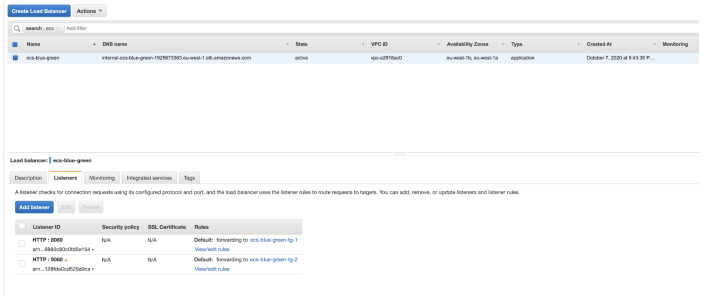


Immagine Docker

Iniziamo descrivendo il *building-block* fondamentale per la nostra architettura a micro-servizi: un'immagine Docker. In questo step potete utilizzare una vostra immagine di preferenza, o una su cui state già lavorando per il vostro progetto. Nel nostro caso, al fine del tutorial, abbiamo realizzato una semplice applicazione Node.js per servire un web server (utilizzando Express) in ascolto sulla porta 3000 che serve 2 API.

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))
app.get('/health', (req, res) => res.send('Health check status: ok'))

app.listen(3000, () => console.log(`Example app listening at http://localhost:${port}
`))
```

Come potete notare, abbiamo esposto un API di health check sotto la route *health.* Se state usando una vostra immagine Docker, è necessario che inseriate questa, o un'altra rotta di health check, in quanto sarà uno step fondamentale in fase di deployment per valutare se la nostra nuova versione software sia "healthy" o meno.

Come tutte le applicazioni Node.js, non può mancare il file package.json:

```
 {
  "name": "ecs-blue-green",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Perfetto, il nostro micro-servizio è pronto! Non ci resta che fare la build dell'immagine Docker e salvarla su AWS. Anche qui, per scopo puramente dimostrativo, utilizziamo un Dockerfile molto

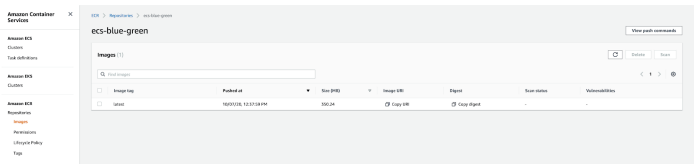
semplice (esatto, i più esperti sanno bene che l'immagine creata potrebbe essere notevolmente snellita!):

```
FROM node:10

COPY . .
RUN npm install

EXPOSE 3000
CMD [ "node", "app.js" ]
```

Push immagine Docker su ECR

Creiamo un repository su Elastic Container Registry (ECR) per ospitare le nostre immagini Docker. Attraverso la Console l'operazione è molto semplice. Aprendo il repository appena creato possiamo accedere alla sezione "View push commands" per ottenere i comandi necessari a buildare e pushare la nostra immagine Docker su ECS.

Ad operazione conclusa, dovreste vedere questo in Console:

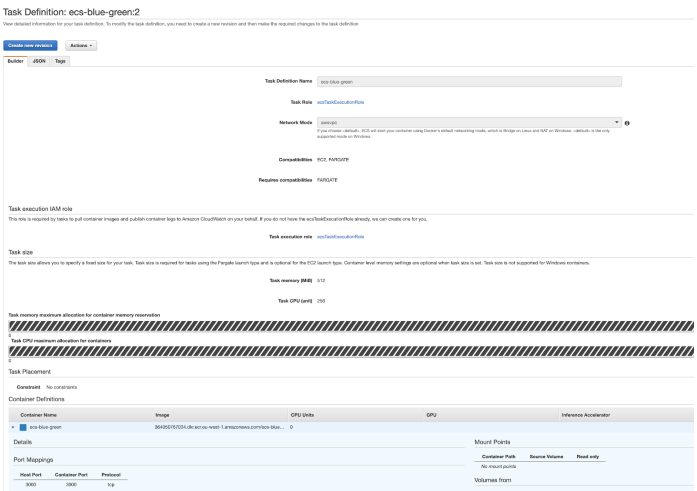


Configurazione ECS – Fargate

A questo punto dobbiamo configurare la Task Definition, l'unità di calcolo fondamentale di ECS che ospiterà i nostri container Docker.

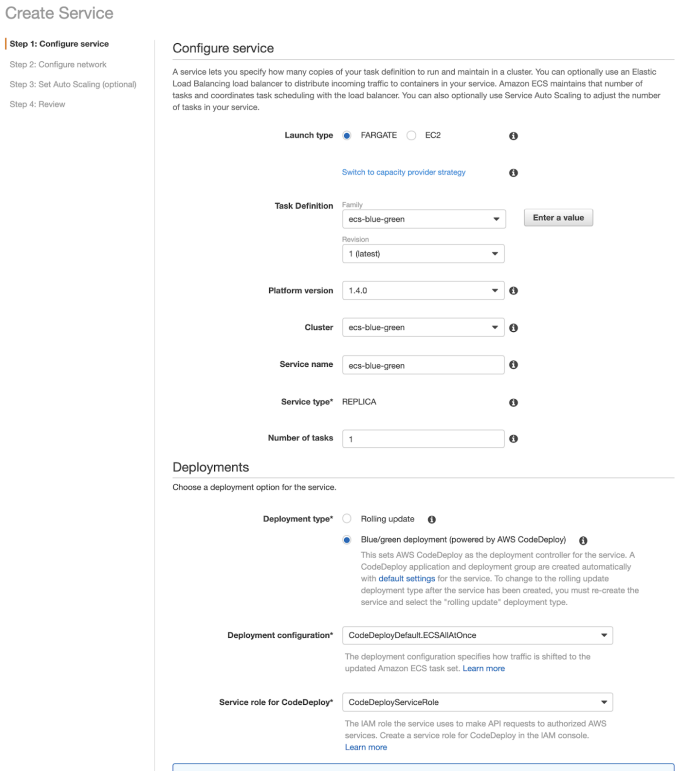
Per ora possiamo assegnare i ruoli di default sia al *Task Role* che al *Task Execution Role* dal momento che sono sufficienti per le operazioni che dobbiamo svolgere. Successivamente creiamo un container e lo associamo all'immagine Docker, precedentemente salvata su ECR, configurando opportunamente la vCPU e la memoria da riservare. Non dimenticate di esporre la porta 3000 dal container!

Il risultato finale della Task Definition appena creata sarà come segue:



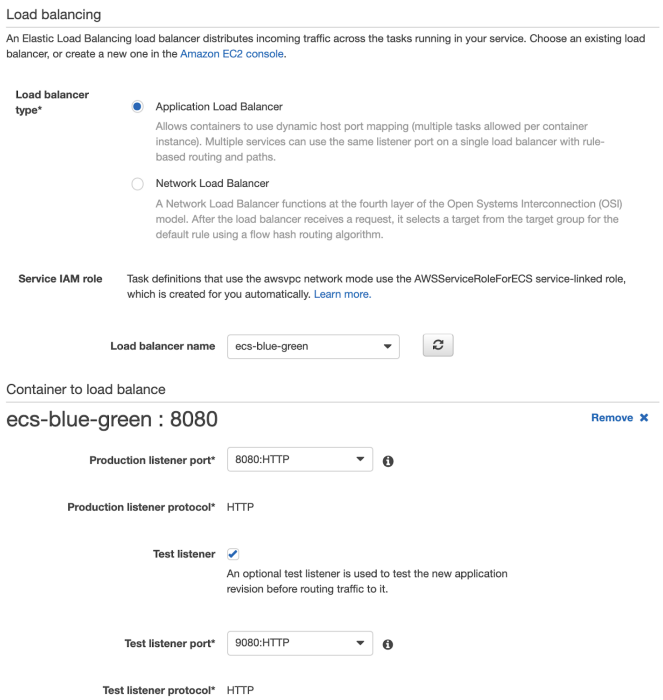
Concludiamo questa fase con la creazione del Service.

Procedendo tramite Console, nella prima pagina è necessario selezionare “blue/green deployment”. Fate attenzione a questo passaggio, nel caso in cui doveste selezionare “Rolling update” dovrete cancellare e ricreare il Service siccome l’opzione non sarà più modificabile:



Nelle sezioni successive potete inserite i dati delle vostre VPC, subnet e security group.

è il momento di utilizzare l’Application Load Balancer creato in fase di definizione dei prerequisiti. CodeDeploy ci richiede infatti di indicare l’Application Load Balancer, i Listener e i Target Group da utilizzare per la gestione del blue/green deployment.

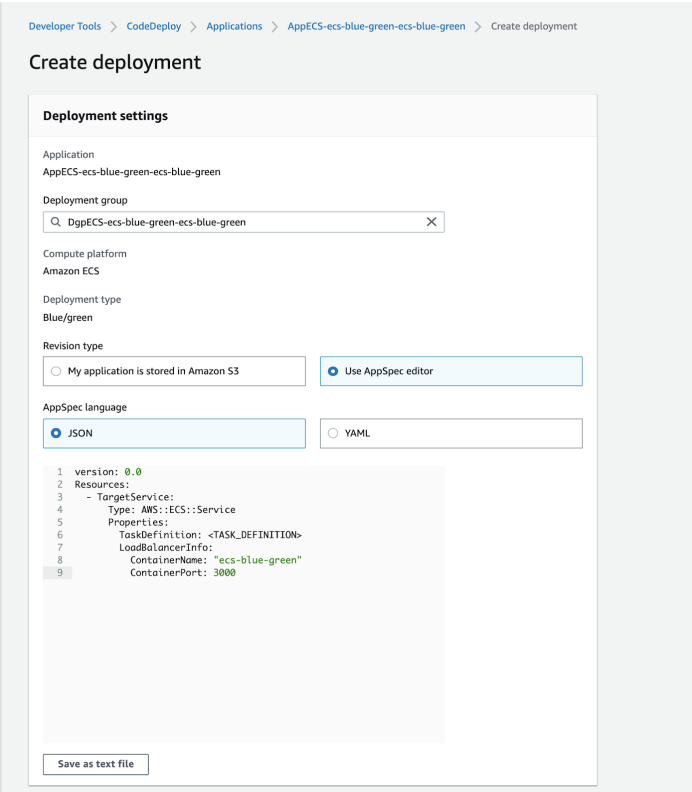


Deploy

Bene, non ci resta che provare un deploy!

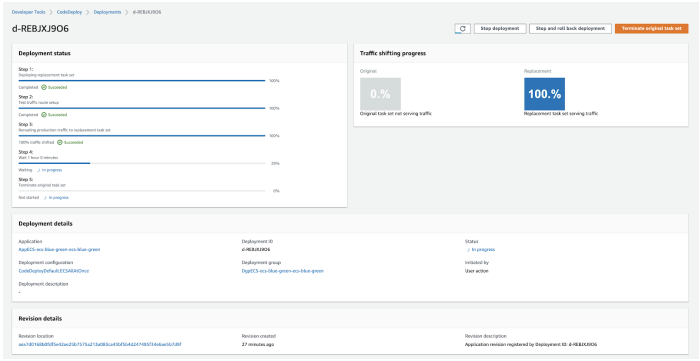
Aprendo la pagina di CodeDeploy notiamo che è stata creata un'Applicazione insieme ad un Deployment Configuration Group. Questi componenti sono già stati configurati per noi in fase di creazione del service ECS. Per poter avviare un deployment è necessario fornire un file yaml con una sintassi ben precisa, consultabile sulla documentazione di AWS(https://docs.aws.amazon.com/codedeploy/latest/userguide/reference-appspec-file.html).Nel nostro caso, andiamo a definire nel file *appspec.yaml* l'entry point del Service ECS insieme alla versione della *task definition* da utilizzare:

```yaml
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition:
        LoadBalancerInfo:
          ContainerName: "ecs-blue-green"
          ContainerPort: 3000
```



Ricordatevi di sostituire <TASK_DEFINITION> con l'arn del task definition che volete deployare.

Completata la configurazione, si atterra sulla pagina del deployment. Qui potete monitorare il suo andamento. Una volta che il nuovo task sarà installato, vedrete una schermata simile a questa:



Su CodeDeploy potete monitorare l'andamento del vostro deployment, capire se ci sono problemi durante l'installazione del nuovo pacchetto software, analizzare la quantità di traffico sulla vecchia e sulla nuova infrastruttura, oltre ad avere la possibilità di effettuare un rollback qualora lo riteneste opportuno.

A questo punto il blue/green deployment è terminato. Potete decidere se effettuare rollback (se magari vi siete accorti che il nuovo applicativo non performa in modo corretto) o approvare direttamente la nuova versione del software.

Potete anche affidarvi totalmente al vostro sistema di rilascio senza dover promuovere o fare rollback manualmente. In questo caso, l'infrastruttura "blue" verrà automaticamente promossa, in caso di installazione andata a buon fine, dopo il tempo che avete configurato voi.

Conclusioni

In questo articolo abbiamo visto come creare un Service ECS in grado di gestire rilasci in modalità blue/green. La configurazione presentata può essere agevolmente sostituita da una pipeline triggerata direttamente dal nostro sistema di versionamento GIT. In poche parole, da un semplice *git push* possiamo rilasciare in modalità blue/green il nuovo pacchetto software.

Volete scoprire come si fa? Seguiteci: prossimamente vedremo insieme come configurare una pipeline per gestire l'intero flusso di rilascio.

#Proud2beCloud vi da appuntamento a tra 14 giorni.

Arrivederci al prossimo deploy!

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it