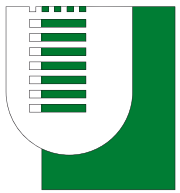
AWS Elastic Beanstalk, Simple Queue Service

Hands-on Cloud Computing Services

Gabriele Russo Russo



TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Ansible: una precisazione

- Ansible può essere usato per configurare host Windows
- Ansible NON può essere installato su Windows
  - I playbook possono essere avviati da una VM Linux

Photogallery

Fino a qui abbiamo utilizzato:

- EC2 (con Load Balancer, Auto Scaling, VPC)
- S3
- DynamoDB
- CloudFront

Aggiungiamo le seguenti feature:

- Ridimensionamento automatico delle immagini in fase di upload
- Applicazione di filtri in fase di upload
- **Problema**: operazioni "pesanti" da eseguire sul web server...

Vediamo come semplificare il deployment dell'applicazione e renderla più scalabile con **AWS Elastic Beanstalk** e **SQS**.

Elastic Beanstalk

Elastic Beanstalk is an ***easy-to-use service for deploying and scaling web applications and services*** *developed with Java, .NET, PHP, Node.js, Python, Ruby, Go [. . .]*

You can ***simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling*** *to application health monitoring.*

- PaaS (?)
- Pricing[1]

[1] `https://calculator.s3.amazonaws.com/index.html`

Photogallery con Beanstalk

Servono alcuni accorgimenti per effettuare il deployment della nostra applicazione web:[2]

- file `requirements.txt`: lista di tutte le dipendenze
- rinominiamo il file principale: `application.py`
- l'applicazione deve avere le credenziali per utilizzare le API di AWS (per S3 e DynamoDB)
  - nelle impostazioni di sicurezza, dobbiamo associare al ruolo `aws-elasticbeanstalk-ec2-role` le policy necessarie (e.g., *AmazonS3FullAccess*)
- `photogallery_v5`

[2] `https://docs.aws.amazon.com/it_it/elasticbeanstalk/latest/dg/create-deploy-python-flask.html`

SQS

Amazon Simple Queue Service (SQS) è un servizio di accodamento messaggi completamente gestito che consente la separazione e la scalabilità di microservizi, sistemi distribuiti e applicazioni serverless. SQS permette di eliminare la complessità e il sovraccarico associati con la gestione e il funzionamento di middleware orientato ai messaggi e consente agli sviluppatori di concentrarsi sulla differenziazione del lavoro.

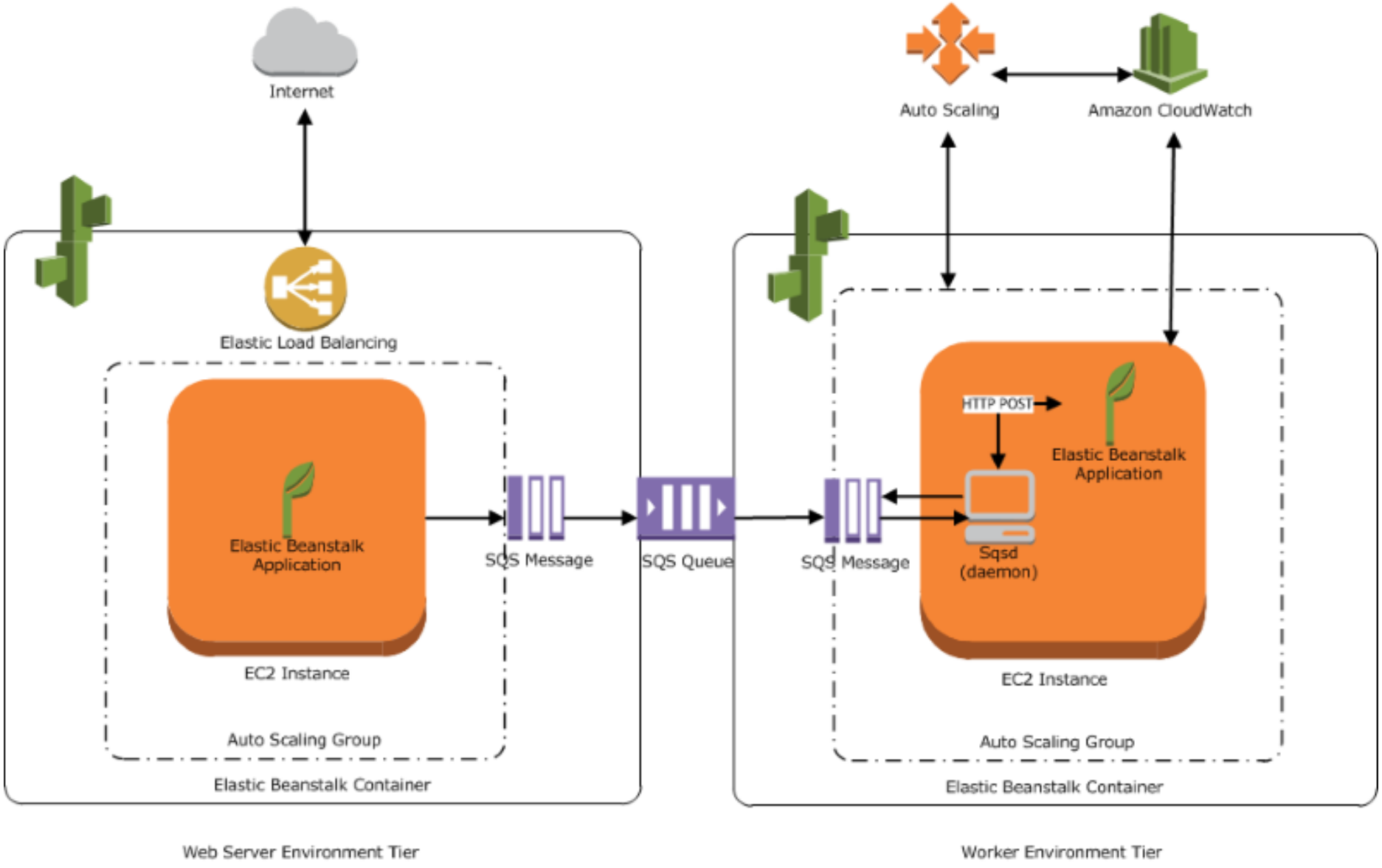
- Code standard (at-least-once)
- Code FIFO (exactly-once, ordinamento FIFO)

- `example_sqs.py` (producer) e `example_sqs2.py` (consumer)

Photogallery con SQS

- Il web server in fase di upload carica l'immagine su S3 nella directory `pending/`
- Processamento delle immagini delegato ad un componente *worker*
- Il web server invia un messaggio ad una coda SQS indicando la chiave dell'immagine su S3 e il filtro da applicare
- Un server worker legge il messaggio dalla coda e completa il processamento

Beanstalk + SQS



Photogallery con Beanstalk+SQS

Su **Beanstalk** ambiente **worker** ad hoc:

- associato ad una coda SQS
- un demone invia una richiesta HTTP al worker per ogni messaggio recuperato dalla coda
- per l'ambiente worker specifichiamo tra le impostazioni:
  - file principale `worker.py`
  - chiediamo di ricevere i messaggi in codifica `text/plain` nel corpo delle richieste HTTP
- `photogallery_v6`

Photogallery: sviluppi futuri

- Aggiunta di commenti alle immagini: dove memorizzarli?
- Ricerca di immagini per titolo o tag: come farlo in maniera efficiente?
- Miglioramento del processo di upload:
  - il frontend attualmente conferma l'avvenuto caricamento appena viene inviato il messaggio su SQS
  - i metadati su DynamoDB dovrebbero essere inseriti solo se il processamento dell'immagine va a buon fine
- Salvataggio di versioni diverse delle immagini (e.g., a bassa risoluzione per la home, ad alta risoluzione per visualizzazione del singolo post)

Esercizio: inviate la vostra versione della'applicazione con una breve descrizione dell'architettura e dell'infrastruttura cloud utilizzata.