# EC2 Autoscalling

Les groupes ne sont pas configurés pour scaller selon la charge et pourraient souvent utiliser une plus petite machine en dev/uat.

À vue rapide et en regardant seulement les groupes de (2+ / large+) machines. Les suivants pourraient être ajustés. J’ai aussi appercu un cluster Kafka dans un autre compte qui semblait aussi très grave.

Identiques en dev/uat (2x) (max sur 1 semaine, moyenne beaucoup plus basse) économies

* activemq (2 x t3.medium @ max: 14% MEM)
* impala-master (1 x t3.large @ max: 18% MEM)
* impala-worker (5 x r5.large @ max: 12% MEM)
* airflow-worker (1 x m5.large @ max: 20% MEM)
* dev-hive (1 x c5.large @ max: 26% MEM)
* s3transfer (1 x t3.xlarge @ max: 23% MEM)
* s3transfer-consumer-daily (1 x t3.xlarge @ max: 7% MEM)

Différents en dev/uat

* spark-master (dev) (1 x m5.xLarge @ max: 9% MEM)
* spark-master (uat) (1 x m5.2xLarge @ max: 17% MEM)

### Dynamic scalling policies

* Les groupes d’autoscalling n’ont pas de règles qui augmentent ou réduisent selon la charge.
* Ils ont toutefois une règle qui les fermes la fin de semaine et la nuit (bon)

Graphical user interface, text, application

Description automatically generated

$0.0116 per On Demand Linux t3.micro

$0.0232 per On Demand Linux t3.small

$0.0464 per On Demand Linux t3.medium

$0.0928 per On Demand Linux t3.large

$0.093 per On Demand Linux c5.large

$0.107 per On Demand Linux m5.large

$0.1856 per On Demand Linux t3.xlarge

$0.214 per On Demand Linux m5.xlarge

$0.276 per On Demand Linux r5.xlarge

$0.428 per On Demand Linux m5.2xlarge

# MQ (AmazonMQ / ActiveMQ)

### Instance sizes

Les environnements de dev et uat pourraient utiliser les instances *mq.t3.micro* spécialement créées pour les environnements de dev. Les instances *mq.t3.micro* pourraient possiblement être amenées en prod pour certain tenants (à étudier).

Économie : 578$/mois (90%).

*Les instances actuelles sont littéralement inutilisées.*

Text

Description automatically generated with medium confidence

* Aucun message dans aucune queue à l’exception de deux (DLQ.AIRFLOW.FILEDETECTION et SYSTEM.EVENT\_QUEUE) qui ne semble pas être consumées (devraient être vidées).
* Memory usage et CPU ne dépassent jamais 5%

### Duplication de services

Il y a des instances nommées « *[tenant]-[zone][env]-activemq »*. Elles pourraient possiblement utiliser le service.

# CloudWatch (CW Agent)

50% des coûts associés proviennent des coûts de métriques personnalisées (0.30$ / métrique / instance) générées par l’agent CloudWatch.

Couts actuels

Métriques personnalisées: $363 / mois (1,211 métriques)

Log Events : $350 / mois (635GB) (Les granularité seraient à vérifier)

### Métriques dupliquées ou calculables (Métriques personnalisées)

Certaines métriques sont supposées être calculées dans les queries CloudWatch (min, max, avg, etc…). Elles ont effectivement la même valeur.

Example

Connections / connections\_min / connections\_max

### Sélection des métriques

Beaucoup de métriques ne sont pas nécessaires ou sont le résultat d’un sélecteur étoile \*.

Tous les volumes surveillés (65%)

|  |  |
| --- | --- |
| /  /sys/fs/cgroup  /run  /dev/shm | /dev  /tmp  /run/user/997 |

### Métriques par instances

Toutes les métriques sont répliquées par instance, mais beaucoup n’en n’ont pas besoin ou profiteraient même à être enregistées au niveau du groupe d’autoscalling ou de l’index de l’instance dans le groupe (0, 1, 2, 3…).

En résumé, le fichier de configuration contient le bloc suivant (qui sapplique à toutes les métriques)

Text, letter

Description automatically generated

Exemples

Requêtes http - (endpoints, connections, etc…).

Beaucoup d’autres

Ressources :

[support pour plusieurs fichiers de configuration](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Agent-common-scenarios.html#CloudWatch-Agent-multiple-config-files)

Notes

L’utilisation de [namespaces](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_concepts.html#Namespace) clarifierait énormément la navigation dans CloudWatch.