Industrialization of IaaS Platform

- EN
- FR

You can use tags to differentiate resources from one another. Adding tags enables you to provide contextual metadata for your resources and organize them efficiently.

This topic provides best practices to make the most of tags and metadata.

Related Pages

- Tagging Your Resources
- Using Metadata and User Data
- Configuring an Instance with User Data and 3DS OUTSCALE Tags
- Creating an Autonomous Instance

Adding Tags to Instances (VMs)

With the 3DS OUTSCALE Cloud, you can use property tags on different resources for effective organization. Adding tags to different resources will allow you to identify and categorize resources. For more information, see Adding or Removing Tags.

We recommend creating groups of tags to organize your resources. Companies that are the most efficient in their use of tags usually create tag groupings that enable them to organize their resources along technical, business, and security dimensions. You can also create automation-specific tags to support your automation efforts.

Tag Examples

Technical Tags	Description
Name	Tag to identify resources.
Environment	Tag to identify the environment, as there are usually multiple environments (production, integration, test, and so on).
Version	Tag to identify the version of the resources.

Business Tags	Description
Owner	Tag to identify the person responsible for the resource.
Customer	Tag to identify a specific customer that a particular group of resources serves.
Project	Tag to identify the project the resource supports.

Security Tags	Description
Confidentiality	Tag to identify the level of data confidentiality of the resource.

Compliance	Tag to identify compliance requirements.

Automation Tags	Description
Date/Time	Tag to identify the date or time a resource should be started, stopped, deleted, or rotated.
Security	Tag to determine security requirements, such as encryption.

Using Metadata to Configure Your Instance (VM)

You can get information about your instances. For more information, see Accessing the Metadata and User Data of an Instance.

This information is very useful when writing scripts which are executed within the instance. This works for any 3DS OUTSCALE Linux instance, and the IP address always stays the same.

When launching an instance for the first time, you need to run scripts that initialize and install all the tools you need. These scripts will require information about your instance, and this information can be found in the metadata.

Example

```
User $> curl 169.254.169.254/latest/meta-data
/instance-id
i-be1dcf1a
```

Using User Data to Get Your Tags

In order to automate configuration deployment, you can use user data to store and run scripts at every boot and get all the tags of your instance. For more information, see Creating an Autonomous Instance.

Below is an example of how you can get all the tags related to your instance:

Getting All Tags of Instance with instanceId

```
User $> aws ec2 describe-instances --instance-
id <INSTANCE-ID> --endpoint <ENDPOINT> | jq .
Reservations[].Instances[].Tags[].Value
"First VM 2 With Perso Snap"
"servitude"
"VmFromPersoOmi"
```

Thanks to the tags you get in the response, you will be able to customize your deployment. Thus, you can use tags to manage your local system and your infrastructure.

Using Tags of an Instance to Apply a Specific Configuration

Before you begin: Configure the tags when you create the instance, and get the ID of the instance.

1. Get the tags of the instance:

```
User $>aws ec2 describe-instances --
instance-id i-819fbfd7 | jq '.Reservations
[].Instances[].Tags[] | select(.Key=="
APT") | .Value' --raw-output
nginx htop tree wget vim
```

2. Depending on your distribution, perform one of the following actions:

```
# For Debian Or Ubuntu Distribution
User $>sudo apt-get -y install nginx htop
tree wget vim
# For CentOS Distribution
User $>sudo yum install -y nginx htop
tree wget vim
```

3. Check the installation and the returned status.

Below is an example of a simple script to automate the installation of packages with APT tags:

Automating the installation of packages with APT tags

```
#/bin/sh
ip_new_vm=$(aws ec2 describe-instances --
instance-id i-d7d895d8 | jq '.Reservations[].
Instances[].PublicDnsName' --raw-output)
list_pkg=$(aws ec2 describe-instances --
instance-id i-d7d895d8 | jq '.Reservations[].
Instances[].Tags[] | select(.Key=="APT") | .
Value ' --raw-output)
ssh user@${ip_new_vm} sudo apt-get install
${list_pkg}
ssh user@${ip_new_vm} dpkg -l | egrep -i
'nginx|htop|tree|wget|vim' | wc -l
```

Customer Use Cases

In the following use cases, you can see how you can use tags in automated scripts.

Starting/Stopping an Instance (VM)

You can run automated start/stop scripts that turn off development environments during non-business hours in order to reduce costs. Using tags enables you to easily identify instances specifically used for development.

If you manage your resources well, you can plan when to start and stop your items directly in your crontab.

- To get a list of specific instances, for example, instances tagged with *Development*, and stop them at 7 p.m.:

Stopping Instances

```
User $> aws ec2 describe-instances --
filter "Name=tag:Type,Values=DEV" | jq '.
Reservations[].Instances[].InstanceId' --
raw-output > /tmp/lst.vm.Dev
i-aff8855f
i-819fbfd7
User $> for vm in `cat lst.vm.Dev` ; do
echo $vm ; aws ec2 stop-instances --
instance-ids $vm ; done
```

- To start the instances in the morning, at 7 a.m.:

Starting Instances

```
User $> for vm in `cat lst.vm.Dev` ; do
echo $vm ; aws ec2 start-instances --
instance-ids $vm ; done
User $> rm /tmp/lst.vm.Dev
```

- To finalize the automation, you need to add scripts (StopInstanceDev.sh and StartInstanceDev.sh) in Crontab. Here is an example of a simple cron job:

Cron Job

```
# Running Script Every Day at 19h
00 19 0 0 0 . /home/user/.profile ; sh
/home/tools/StopInstanceDev.sh
# Running Script Every Day at 07h
00 07 0 0 0 . /home/user/.profile ; sh
/home/tools/StartInstanceDev.sh
```

Deleting Old Snapshots

If you use snapshots to back up instances, you can easily forget to delete old snapshots that have become useless. In order to avoid this, you can automate the deletion of useless snapshots thanks to tagging.

- To get all snapshots older than 2017-10:

Geting All Snapshots Older Than

```
User $> aws ec2 describe-snapshots --
query 'Snapshots[?StartTime>=`2017-10-
01`].SnapshotId' --output text > /tmp/lst.
snap.older.10.2017
```

- To delete useless snapshots:

Deleting Snapshots

```
User $> snapshots_to_delete=`cat lst.snap.
older.10.2017` ; for snap in
$snapshots_to_delete ; do aws ec2 delete-
snapshot --snapshot-id $snap ; done
User $> rm /tmp/lst.snap.older.10.2017
```

As a good practice, we recommend putting version tags on all your resources. You can also tag instances with *Development*, *Integration* and *Production*. With this type of tag, you can apply SALT recipe automatically for instances with the *Development* profile.

Conclusion

We strongly recommend adding tags to your resources, especially the most expensive ones of your account. Ultimately, tagging is the best way to follow your consumption. It will make your resources easier to navigate, and improperly configured resources easier to spot. Tagging is a critical tool in your management toolkit and the best way to keep a clear view of your Cloud resources.