AWS Compute Blog

Automating the Creation of Consistent Amazon EBS Snapshots with Amazon EC2 Systems Manager (Part 1)

by Bryan Liston | on 07 MAR 2017 | in Amazon EC2 | Permalink | Comments | Share

Nicolas Malaval, AWS Professional Consultant

If an EC2 instance is up and running, there may be applications working, like databases, with data in memory or pending I/O operations that cannot be retrieved from an Amazon EBS snapshot. If your application is unable to recover from such a state, you might lose vital data for your business.

Amazon EBS provides block level storage volumes for use with EC2 instances. With EBS, you can create point-in-time snapshots of volumes, stored reliably on Amazon S3. If you rely on EBS snapshots as your backup solution and if you cannot turn off the instance during backup, you can create consistent EBS snapshots, which consists of informing the applications that they are about to be backed up so they can get prepared.

In this post, the first of a two-part series, I show you how to use Run Command and Maintenance Window, two features of Amazon EC2 Systems Manager, to automate the execution of scripts on EC2 instances that create consistent EBS snapshots. First, I explain the approach. Then, I walk you through a practical example to create consistent snapshots of an Amazon Linux EC2 instance running MySQL.

Creating consistent EBS snapshots with Run Command

Run Command lets you securely and remotely manage the configuration of Windows or Linux instances. For example, you can run scripts—or commands—without having to log on locally to the instance. Run Command requires the SSM Agent to be installed on the EC2 instances.

I use Run Command to run a script remotely on EC2 instances. The script coordinates the preparation of applications and the creation of EBS snapshots, as follows:

1. It instructs the applications and the file system to flush their cached data to the disk and then to temporary block all I/O operations. At this moment, the EBS volume is in a consistent state.
2. It retrieves the ID of the instance running the script using the Instance Metadata.
3. It queries the EC2 API to obtain the ID of the EBS volumes attached to the instance, then to create a snapshot of each of the EBS volumes.
4. Finally, it thaws I/O operations as soon as the EC2 API responds to the last request with a snapshot ID. It is not necessary to wait for the snapshot to complete.

The content of the script varies upon the system and the applications that should be prepared for backup. See the example sections later in this post.

Instances communicate with the Run Command API to retrieve commands to execute and return results, and with the EC2 API to get volume attachment information and create EBS snapshots. To grant permission to call the APIs, I launch the instances with an IAM role for EC2 instances. This role is attached to the SSM Managed Policy **AmazonEC2RoleforSSM** and to an inline policy which allows ec2:DescribeInstanceAttribute and ec2:CreateSnapshot actions.

Using Run Command has multiple benefits:

- The scripts are maintained centrally and any changes are effective immediately on every instance
- Commands are executed remotely and the instances continuously retrieve and run new commands
- Status and results of each command execution are reported by Run Command and the information is also stored in AWS CloudTrail for audit purposes
- Run Command is integrated with IAM to allow you to control both the users and level of access

Executing commands on a daily basis with Maintenance Windows

Maintenance Windows allows you to specify a recurring time window during which Run Command tasks are executed. I use Maintenance Windows to create consistent EBS snapshots on a daily basis during off-peak hours, because it may temporarily increase resource utilization and affect application performance.

The maintenance window is registered with multiple targets. Each target is a set of EC2 instances that have a tag “ConsistentSnapshot” assigned and an arbitrary value depending on what script to execute. Each target is registered with a task assigned to an SSM document, which describes the actions to perform by Run Command to create consistent EBS snapshots on every instance of this target.

Automating the creation of consistent EBS snapshots of an Amazon Linux instance running MySQL

Here’s a practical example to create consistent EBS snapshots of an Amazon Linux instance running MySQL, with step-by-step instructions.

Understanding the example

I use Run Command to execute a shell script on the Amazon Linux instance:

```
Bash
mysql -u backup -h localhost -e 'FLUSH TABLES WITH READ LOCK;'
```

First, the shell script prepares MySQL for backup. The command FLUSH TABLES WITH READ LOCK waits for the active transactions to complete, flushes the cache to the filesystem, and prevents clients from making write operations (see FLUSH in the MySQL documentation). You should note that this MySQL backup method implies a short interruption of write operations, and the duration depends on the current size and workload. You should make sure that the backup does not affect your applications.

```
Bash
sync

for target in $(findmnt -nlo TARGET -t ext4); do fsfreeze -f $target; done
```

It then suspends access to the filesystems and creates a stable image
on disk. At this stage, the EBS volume is in a consistent state.

```
Bash
instance=`curl -s http://169.254.169.254/latest/meta-data/instance-id`
region=`curl -s 169.254.169.254/latest/meta-data/placement/availability-zone`
region=${region::-1}
volumes=`aws ec2 describe-instance-attribute --instance-id $instance --attribute blockDevi

for volume in $(echo $volumes | tr " " "\n")
do aws ec2 create-snapshot --volume-id $volume --description 'Consistent snapshot of MySQL
done
```

It creates a snapshot of every EBS volume attached to the instance.

```
Bash
for target in $(findmnt -nlo TARGET -t ext4); do fsfreeze -u $target; done

mysql -u backup -h localhost -e 'UNLOCK TABLES;'
```

Finally, it resumes access to the filesystems and unlocks MySQL.

This shell script is contained in a new SSM document. The maintenance window executes a command from this document every day at midnight on every Linux instance that has a tag “ConsistentSnapshot” equal to “AmazonLinuxMySQL”.

Implementing and testing the example

First, use AWS CloudFormation to provision some of the required resources in your AWS account.

1. Open Create a Stack to create a CloudFormation stack from the template.
2. Follow the on-screen instructions.

CloudFormation creates the following resources:

- A VPC with an Internet gateway attached
- A subnet on this VPC with a new route table to enable access to the Internet and therefore to the AWS APIs
- An IAM role to grant an EC2 instance the required permissions
- An Amazon Linux instance in the subnet with the IAM role attached and the user data script entered to install and configure MySQL and the SSM Agent at launch
- A SSM document containing the script described in the section earlier.
- An IAM role to grant the maintenance window the required permissions

After the stack creation completes, choose **Outputs** in the CloudFormation console and note the values that the process returned:

- IAM role for the maintenance window
- Name of the SSM document

Manually create a maintenance window:

1. In the Amazon EC2 console, choose **Systems Manager Shared Resources**, **Maintenance Windows**, **Create a Maintenance Window**.
2. For **Name**, enter ConsistentSnapshots.
3. For **Specify with**, choose **CRON/Rate expression.** For **CRON/Rate expression**, enter **cron(0 0 * * ? *)** in. This creates consistent EBS snapshots every day at midnight UTC.
4. For **Duration**, enter 2 hours. For **Stop initiating tasks**, enter 0 hour.
5. Choose **Create maintenance window**. The system returns you to the Maintenance Window page.

After you create a maintenance window, assign a target where the task will run:

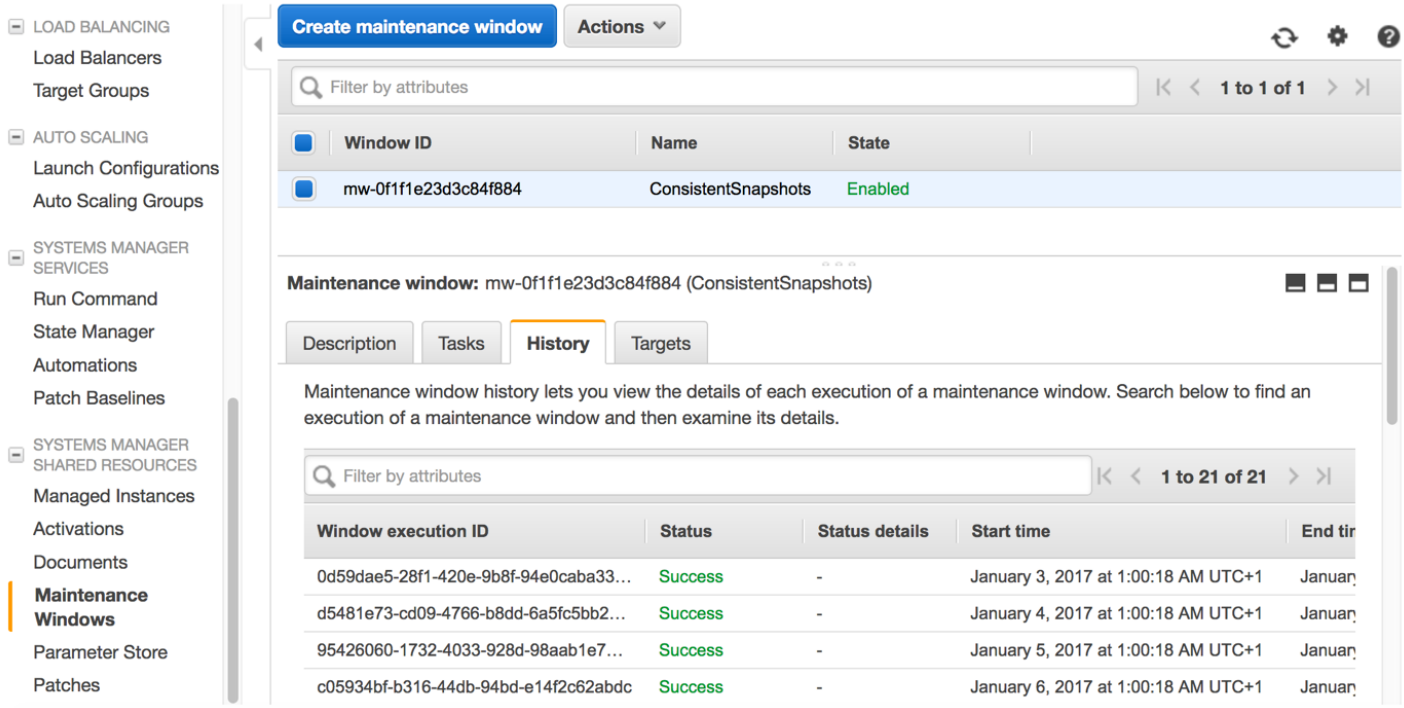
1. In the **Maintenance Window** list, select the maintenance window that you just created.
2. For **Actions**, choose **Register targets**.
3. For **Owner information**, enter AmazonLinuxMySQL.
4. Under **Select targets by** section, choose **Specifying tags**. For **Tag Name**, choose ConsistentSnapshot. For **Tag Value**, choose AmazonLinuxMySQL.
5. Choose **Register targets**.

Finally, assign a task to perform during the window:

1. In the **Maintenance Window** list, select the maintenance window that you just created.
2. For **Actions**, choose **Register tasks**.
3. For **Document**, select the SSM document that was returned by CloudFormation.

4. Under the **Target by** section, select the target that you just created.
5. Under the **Role** section, select the IAM role that was returned by CloudFormation.
6. Under the **Execute on** section, for **Targets**, enter 1. For **Stop after**, enter 1 errors. You can adapt these numbers to your own needs.
7. Choose **Register task**.

You can view the history either in the **History** tab of the Maintenance Windows navigation pane of the Amazon EC2 console, as illustrated on the following figure, or in the **Run Command** navigation pane, with more details about each command executed.



Conclusion

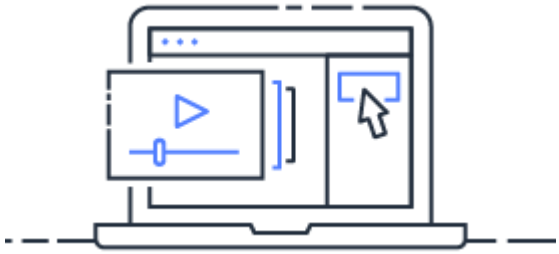
In this post, I showed how you can use Amazon EC2 Systems Manager to create consistent EBS snapshots on a daily basis, with a practical example for MySQL running in an Amazon Linux instance.

In the next part of this two-part series, I walk you through another example to create consistent snapshots of a Windows Server instance with Microsoft VSS (Volume Shadow Copy Service).

If you have questions or suggestions, please comment below.

AWS Podcast
Subscribe for weekly AWS news and interviews
Learn more »

AWS Partner Network
Find an APN member to support your cloud business needs
Learn more »



AWS Training & Certifications
Free digital courses to help you develop your skills
Learn more »

Resources

Serverless Computing and Applications
Amazon Container Services

AWS Messaging
Cloud Compute with AWS
Desktop and Application Streaming

Follow

Twitter
Facebook
LinkedIn
Twitch
Email Updates

AWS Events
Discover the latest AWS events in your region
Learn more »

Related Posts

How AWS Game Tech helped three indie devs launch a game-as-a-service

Setting up Grafana on EC2 to query metrics from Amazon Managed Service for Prometheus

Running cost optimized Spark workloads on Kubernetes using EC2 Spot Instances

How to monitor Windows and Linux servers and get internal performance metrics

ZeroLight, AWS, and Lucid create a whole new way to buy cars

Configuring and using Oracle Connection Manager on Amazon EC2 for Amazon RDS for Oracle

re:Invent 2020 – Healthcare Industry Recap

Create immutable servers using EC2 Image Builder and AWS CodePipeline