Creating a CHR instance in AWS EC2

RouterOS CHR instances in AWS EC2

I have experience with T2, T3, and T3a, primarily nano and micro, CHR instances. I prefer to use T3 instances as they provide two CPU cores for practically the same low price.

TL;DR

If you want to use T3 or T3a instance, you need to create fresh Router OS 7 CHR instances or else you will experience a corrupted disk on subsequent upgrades.

I have not experienced any upgrade failures with T2 instances.

The whole story

Mikrotik provides a RouterOS CHR v6.44.3 AMI in the AWS Marketplace. I have a number of production EC2 instances built from that AMI. When I started testing ROS 7 instances, I spun up new instances using this AMI, upgrading to the latest v6, and then whatever the current v7 was at the time. These upgrades all succeeded and ran with no issues, until Mikrotik released to the next version and I attempted to upgrade. The result each time, regardless of the target version (7.2, 7.3, 7.4, 7.5, and 7.6), was a corrupted disk. If you view the screenshot of the instance, you will find the last message "Loa01" on the console.

I opened ticket SUP-77812 with Mikrotik on March 24th after experiencing my third router failure. This failure occurred while upgrading from ROS 7.1.3 to 7.1.5.

During some testing in May, I was able to determine that this disk corruption was consistent with T3 instances, but never occurred with T2 instances.

I attempted to create my own AMI with the CHR VHD, VHDX, and OVA disks, but I never was successful. Recently I learned where I went wrong. I was using the "aws ec2 import-image" command., which always resulted in a failure related to missing OS files. I later learned that it

appears this command can only be used to import certain approved operating systems. I should have been using the "aws ec2 import-snapshot" command which will import disk images.

I now have my own AMIs to test with.

Instance Testing

Here are the tests that I performed with t3.nano and t3a.nano instances.

Mikrotik 6.44.3 AMI -> 6.49.7 -> 7.1 -> 7.2 FAILED with disk corruption.
Mikrotik 6.44.3 AMI -> 6.49.7 -> 7.2 -> 7.3 FAILED with disk corruption.
Mikrotik 6.44.3 AMI -> 6.49.7 -> 7.3 -> 7.4 FAILED with disk corruption.
Mikrotik 6.44.3 AMI -> 6.49.7 -> 7.4 -> 7.5 FAILED with disk corruption.
Mikrotik 6.44.3 AMI -> 6.49.7 -> 7.5 -> 7.6 FAILED with disk corruption.

My 6.49.7 AMI -> 7.1 -> 7.2 FAILED with disk corruption.
My 6.49.7 AMI -> 7.1 -> 7.3 FAILED with disk corruption.
My 6.49.7 AMI -> 7.1 -> 7.4 FAILED with disk corruption.
My 6.49.7 AMI -> 7.1 -> 7.5 FAILED with disk corruption.
My 6.49.7 AMI -> 7.1 -> 7.6 FAILED with disk corruption.

My 7.1 AMI -> 7.2 -> 7.3 -> 7.4 -> 7.5 -> 7.6 SUCCESS!!! (At least that means no failures...)

I was even able to take this last instance to the current development pre-release (7.7beta9 currently) and back again a few times with no issues.

It appears that any Mikrotik EC2 T3 or T3a instance that started life as a RouterOS 6 device will succeed on the immediate upgrade to any RouterOS 7 version, but any subsequent upgrade to any other RouterOS 7 minor release will result in failure with a corrupted disk.

Proving the theory

I have tested the following with my 7.6 AMI on T3.nano and T3a.nano instances, downgrading and then upgrading the entire ROS 7.X

My 7.6 AMI -> 7.5 downgrade -> 7.4 downgrade -> 7.3 downgrade -> 7.2 downgrade -> 7.1 downgrade -> 7.2 -> 7.3 -> 7.4 -> 7.5 -> 7.6 SUCCESS the whole way!!!

It would appear that something in ROS 6 is resulting in the eventual death of ROS 7 VMs running as AWS EC2 t3 and t3a instances. On the forums, there have been reports of this occurring with other instance types, but I have not taken the time to catalog data outside of my own experiences.

Steps to create an RouterOS CHR 7.6 Amazon Machine Image

Credit for the AMI creation process goes to the clearlinux team. I found their process documented in a gihub issue linked here.

Prior to using this procedure, you will need to have a properly setup aws cli environment that has permissions (IAM Roles) allowing for vm importing and access to your S3 bucket. Here's Amazon's documentation for adding the permissions and roles.

1. Make sure you have a properly configured AWS CLI environment.
2. Download the CHR 7.6 Raw disk image from the Mikrotik Downloads page.
3. Upload the chr-7.6.img file to an S3 bucket.
4. Create the two files shown below.
5. Update your S3Bucket and S3Key in the json file.
6. Run the following command: "bash aws-import-snapshot-mikrotik-chr-76.sh import-snapshot"
7. Grab the value of **ImportTaskId** from the output of the last command.
8. Run the following command: "bash aws-import-snapshot-mikrotik-chr-76.sh monitor-import [**ImportTaskId**]" with the **ImportTaskId** as the last argument.
9. Wait until the Status shows complete, at which time you can press CTRL-C to break the script.
10. Grab the value of **SnapshotId** from the output.
11. Run the following command: "bash aws-import-snapshot-mikrotik-chr-76.sh register-image [**SnapshotId**]" with the **SnapshotId** as the last argument.
12. Grab the value of **ImageId** from the output of the last command. If everything was successful, this is the ID of your AMI.
13. Go to the AWS control panel for EC2 -> Images -> AMIs to find your new Amazon Machine Image. You can now use it to **Launch instance from AMI**.

mikrotik-routeros-chr-76-raw-containers.json

```
{
  "Description": "Mikrotik RouterOS CHR v7.6 RAW",
  "Format": "raw",
  "UserBucket": {
    "S3Bucket": "my-s3-bucket",
    "S3Key": "mikrotik/chr-7.6.img"
```

```
  }
}
```

aws-import-snapshot-mikrotik-chr-76.sh

```
#!/bin/bash

description="Mikrotik RouterOS CHR v7.6"
json_file="mikrotik-routeros-chr-76-raw-containers.json"

JOB="$1"

case $JOB in
  "import-snapshot")
    aws ec2 import-snapshot --description "${description} image" --disk-container file://${json_file}
    ;;

  "monitor-import")
    import_task_id="$2"

    while true; do
      clear
      date
      echo ""
      aws ec2 describe-import-snapshot-tasks --import-task-ids ${import_task_id}
      sleep 10
    done

    #
    #snapshot_id=$(aws ec2 describe-import-snapshot-tasks --import-task-ids ${import_task_id} | grep SnapshotId
awk -F '"'  '{print $4}')
    #
    ;;

  "register-image")
    snapshot_id="$2"

    aws ec2 register-image \
      --name "$description" \
      --description "$description" \
```

```
    --architecture x86_64 \
    --virtualization-type hvm \
    --ena-support \
    --root-device-name "/dev/sda1" \
    --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\", \"Ebs\": { \"SnapshotId\": \"$snapshot_id\"}}]"
    ;;

  *)
    echo "Unknown job type: ${JOB}"
    ;;
esac
```

END

Revision #7
Created 5 December 2022 05:23:31 by bluecrow76
Updated 2 June 2023 05:20:54 by bluecrow76