Slackware as an AWS EC2 Instance

Prerequisites

- You should have already prepared a DomU compatible Slackware install, see the separate guide on how to do that.
- You should have an AWS account with the necessary vmimport role
- You should have awscli configured with credentials such that commands like 'aws s3 ls' work

That's the bare minimum. However it will help a great deal if you've already tried to import one of the supported operating systems so you know what to expect, and all permissions issues are ironed out.

Security

A quick word about security. We will not be making use of the service provided by AWS which furnishes your instance with key pairs, because we won't be installing the agent process that does that. This saves a bit of time for unsupported operating systems like Slackware, but it means you need to sort out how to get access to your instance after it starts. I strongly suggest you

1. Upload a public key to the account you'll use to access your instance, e.g. with ssh-copy-id.
2. Ensure that ssh is configured to only use key access (PermitRootLogin=prohibit-password), which I believe is the default.
3. Reboot and ensure you still have access using your private key.
4. Double-check that you cannot login using the password that would normally work on the VM console for your user.

Within an hour of your instance becoming live it will be logging access attempts by Russian hackers. AWS address ranges are scanned regularly for vulnerabilities precisely because so many users screw up their security. Don't be caught out.

Disk Formats

You now need to consider in what format you'll upload your Slackware DomU hard disk image.

Stream optimised VMDK is the most efficient way to upload your image. This is the VMDK format created when you export to OVF format from VMWare and you will notice it’s much more compressed than normal VMDK files. Don’t upload a standard VMDK file, it won’t work and you’ll waste your time. Also, don’t try to generate stream-optimised VMDK using any open-source utilities. None of them seem to work properly. If you don’t own a VMWare product with the ‘export to OVF’ option, you could try OVFTool. This is a free download from VMWare for registered users, and will take a VMWare VM directory as input. I’m unsure if it requires a registered copy of VMWare to run but it will expect the .VMX and possibly other files which you’ll have to generate somehow to keep it happy if you're not actually using VMWare. When the OVF directory is generated it will be generated containing several files. You can discard all but the .VMDK.

Ensure you've shutdown VMWare completely and removed any DVD-ROMs from the VM before

exporting to OVF or it may fail with a cryptic (e.g. useless) error message as is typical from VMWare.

If you don't have VMWare, and/or don't want to use OVFTool then you should consider .VHD format. This format was used by the ancient Microsoft emulator called VirtualPC, and latterly HyperV. qemu-img will happily generate one of these when you specify the ‘vpc’ output format. VHD is more verbose than stream-optimised VMDK and will end up about double the size, so double the upload time, and higher S3 charges while importing (not that they will amount to much). You can also use VHDX format as AWS also supports that but there is little point as the generated files are even larger than VHD to contain the same information.

You can also use the RAW format, but you'd have to be some kind of lunatic to do that since you'll be waiting forever for your upload. It's not really necessary.

Snapshot import

The snapshot import process is described by Amazon. If you want to have a read of that feel free to follow it instead of my guide below, or use a combination of the two.

Having generated a hard disk image in the appropriate format you will need to put it in an S3 bucket. This is easily achieved in the AWS console. You can keep it private which is the default setting when you click through the dialogs.

Once uploaded, generate a presigned URL for it using the ARN (Amazon Resource Name) of the bucket item, e.g.

```
aws s3 presign s3://slackware/slackware-for-aws-ovf.vmdk
```

Copy the printed (generated) URL somewhere. You have only limited time to use it, so don't go off to bed just yet!

Create file containers.json containing something like this.

```
{
    "Description": “My Server VMDK 1”,
    "Format": "vmdk",
    "Url" : "<URL>"
}
```

Change vmdk to vhd(x) depending on the format you’re using (it’s not detected by the import process). Choose a description to describe your storage. You may want to put a version in there as I’ve done if you make several attempts at this. Put the presigned URL in the Url field, save the file, and then run:

```
aws ec2 import-snapshot --description "slackware-snap-vmdk-1" --disk-
container "file://containers.json"
```

Again, note my description has a version, I found this handy when looking at logs, and checking on tasks etc…

Take a note of the printed ImportTaskId (or any error!!!) after running that command. e.g. import-

snap-02da847264756b9f3

Monitor the task by repeatedly running something like:

```
aws ec2 describe-import-snapshot-tasks --import-task-ids import-
snap-02da847264756b9f3
```

Once this import has completed (should take about half an hour for Slackware disk sets A, AP, D, K, L, N), take a copy of the final status snapshot ID, as we'll need it e.g.:

```
"SnapshotId": "snap-0677963e6c0b64f80",
```

Connect to an Instance

You can now create an instance. Any instance will do, a t2.micro will have 1 CPU and plenty of RAM to boot Slackware. Shut down the instance, disconnect its root drive. You can delete the root EBS volume supplied with that instance if you want because we'll create another one. There's no point in paying for EBS storage you don't need.

You can then use the AWS Console to create an EBS volume from your uploaded snapshot, and attach the EBS volume as the root device of your instance. Make sure you specify the device as /dev/sda, to ensure it will be the root device. I found this slightly confusing because it's actually /dev/xvda in the running instance but there you go.

You should now be able to start that instance, boot Slackware and connect to it using ssh.

Sources

* Originally written by User Bifferos

howtos, ec2, aws