Amazon Machine Images (AMI)
&
Packer

Tejas Parikh (t.parikh@northeastern.edu)

CSYE 6225
Northeastern University

What is Amazon Machine Images (AMI)?

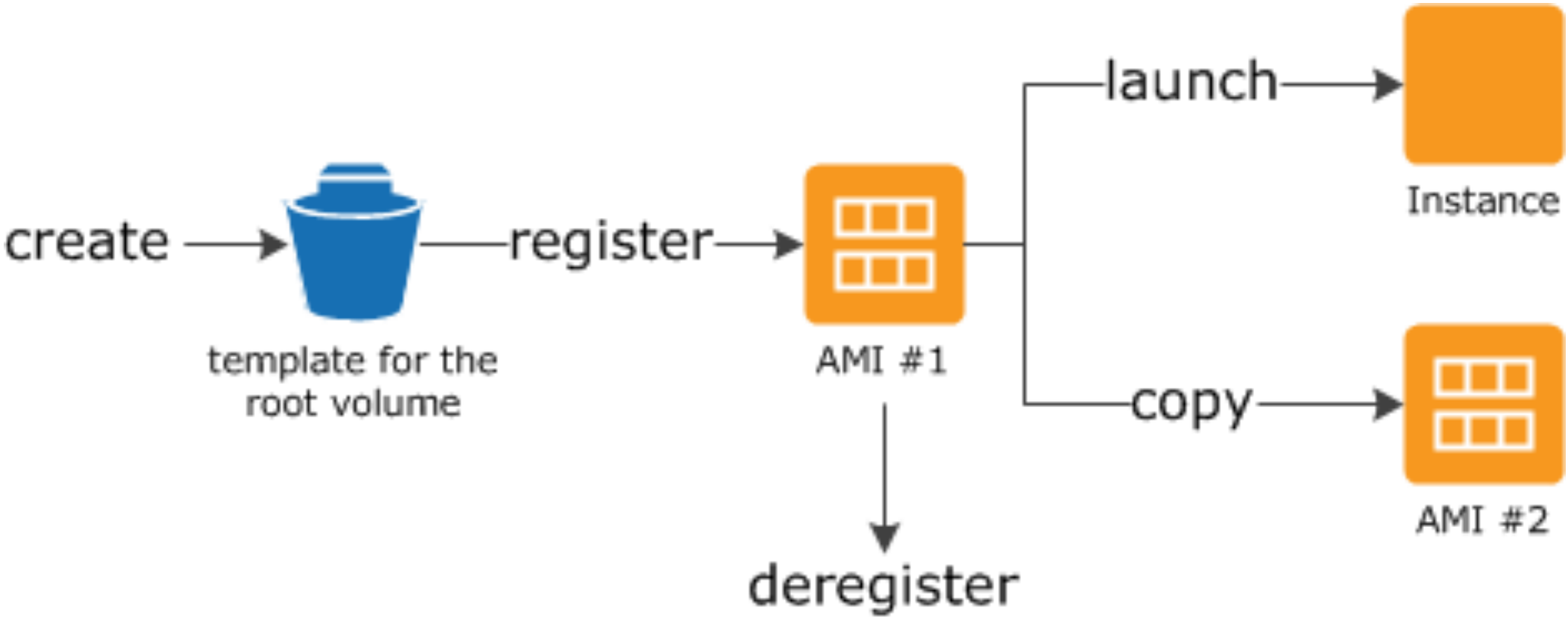
- An Amazon Machine Image (AMI) provides the information required to launch an EC2 instance.
- You must specify a source AMI when you launch an instance.
- You can launch multiple instances from a single AMI when you need multiple instances with the same configuration.
- You can use different AMIs to launch instances when you need instances with different configurations.

What’s in a AMI?

An AMI includes the following:

- A template for the root volume for the instance (for example, an operating system, an application server, and applications).
- Launch permissions that control which AWS accounts can use the AMI to launch instances.
- A block device mapping that specifies the volumes to attach to the instance when it's launched.

AMI Lifecycle



Creating Your Own AMI

- You can launch an instance from an existing AMI, customize the instance, and then save this updated configuration as a custom AMI.
- Instances launched from this new custom AMI include the customizations that you made when you created the AMI.

HashiCorp Packer

- Packer is an open source tool for creating identical machine images for multiple platforms from a single source configuration.

AMI Use Cases

- Golden Images – Secure, immutable OS images are needed especially in Enterprise world where compliance issues matters.
- Environment Parity – Keep all dev/test/prod environment as similar as possible.
- Auto-scaling Acceleration – Launch completely provisioned and configured instances in seconds, rather than minutes or even hours.

Installing Packer

- https://www.packer.io/intro/getting-started/install.html
- Packer may be installed in the following ways:
  - Using a precompiled binary; We release binaries for all supported platforms and architectures. This method is recommended for most users.
  - Installing from source This method is only recommended for advanced users.
  - An unofficial alternative installation method

Build an Image – The Template

- The configuration file used to define what image we want built and how is called a *template* in Packer terminology.
- The format of a template is simple JSON.

Template Structure

- **builders** (required) is an array of one or more objects that defines the builders that will be used to create machine images for this template, and configures each of those builders. For more information on how to define and configure a builder, read the sub-section on configuring builders in templates.
- **description** (optional) is a string providing a description of what the template does. This output is used only in the inspect command.
- **min_packer_version** (optional) is a string that has a minimum Packer version that is required to parse the template. This can be used to ensure that proper versions of Packer are used with the template. A max version can't be specified because Packer retains backwards compatibility with packer fix.
- **post-processors** (optional) is an array of one or more objects that defines the various post-processing steps to take with the built images. If not specified, then no post-processing will be done. For more information on what post-processors do and how they're defined, read the sub-section on configuring post-processors in templates.
- **provisioners** (optional) is an array of one or more objects that defines the provisioners that will be used to install and configure software for the machines created by each of the builders. If it is not specified, then no provisioners will be run. For more information on how to define and configure a provisioner, read the sub-section on configuring provisioners in templates.
- **variables** (optional) is an object of one or more key/value strings that defines user variables contained in the template. If it is not specified, then no variables are defined. For more information on how to define and use user variables, read the sub-section on user variables in templates.

Variables in Packer Template

- The variables section is a key/value mapping of the user variable name to a default value. A default value can be the empty string.
- If the default value is null, then the user variable will be *required*. This means that the user must specify a value for this variable or template validation will fail.

```
{
  "variables": {
    "aws_access_key": "",
    "aws_secret_key": ""
  },

  "builders": [{
    "type": "amazon-ebs",
    "access_key": "{{user `aws_access_key`}}",
    "secret_key": "{{user `aws_secret_key`}}",
    // ...
  }]
}
```

Template Builders

- Within the template, the builders section contains an array of all the builders that Packer should use to generate machine images for the template.
- Builders are responsible for creating machines and generating images from them for various platforms. For example, there are separate builders for EC2, VMware, VirtualBox, etc.
- Packer comes with many builders by default, and can also be extended to add new builders.

AMI Builder (EBS backed)

- The ***amazon-ebs*** Packer builder is able to create Amazon AMIs backed by EBS volumes for use in EC2.
- This builder builds an AMI by launching an EC2 instance from a source AMI, provisioning that running machine, and then creating an AMI from that machine.
- This is all done in your own AWS account.
- The builder will create temporary keypairs, security group rules, etc. that provide it temporary access to the instance while the image is being created. This simplifies configuration quite a bit.
- The builder does not manage AMIs. Once it creates an AMI and stores it in your account, it is up to you to use, delete, etc. the AMI.

Provisioners

- Provisioners use built-in and third-party software to install and configure the machine image after booting.
- Provisioners prepare the system for use, so common use cases for provisioners include:
  - installing packages
  - patching the kernel
  - creating users
  - downloading application code

Various Provisioners

- Ansible Local
- Ansible Remote
- Breakpoint
- Chef Client
- Chef Solo
- Converge
- File
- PowerShell
- Puppet Masterless
- Puppet Server
- Salt Masterless
- Shell
- Shell (Local)
- Windows Shell
- Windows Restart
- Custom

Validating Packer Template

- The packer validate Packer command is used to validate the syntax and configuration of a template.
- The command will return a zero exit status on success, and a non-zero exit status on failure.
- Additionally, if a template doesn't validate, any error messages will be outputted.

```
$ packer validate my-template.json
Template validation failed. Errors are shown below.

Errors validating build 'vmware'. 1 error(s) occurred:

* Either a path or inline script must be specified.
```

Building Packer

- The packer build command takes a template and runs all the builds within it in order to generate a set of artifacts.
- The various builds specified within a template are executed in parallel, unless otherwise specified.
- And the artifacts that are created will be outputted at the end of the build.

```
$ packer build template.json
==> virtualbox: virtualbox output will be in this color.
==> vmware: vmware output will be in this color.
==> vmware: Copying or downloading ISO. Progress will be
==> vmware: Creating virtual machine disk
==> vmware: Building and writing VMX file
==> vmware: Starting HTTP server on port 8964
==> vmware: Starting virtual machine...
==> virtualbox: Downloading VirtualBox guest additions. P
==> virtualbox: Copying or downloading ISO. Progress will
==> virtualbox: Starting HTTP server on port 8081
==> virtualbox: Creating virtual machine...
==> virtualbox: Creating hard drive...
==> virtualbox: Creating forwarded port mapping for SSH (
==> virtualbox: Executing custom VBoxManage commands...
    virtualbox: Executing: modifyvm packer --memory 480
    virtualbox: Executing: modifyvm packer --cpus 1
==> virtualbox: Starting the virtual machine...
==> vmware: Waiting 10s for boot...
==> virtualbox: Waiting 10s for boot...
```

Demo

Additional Resources

https://fall2019.csye6225.cloud/