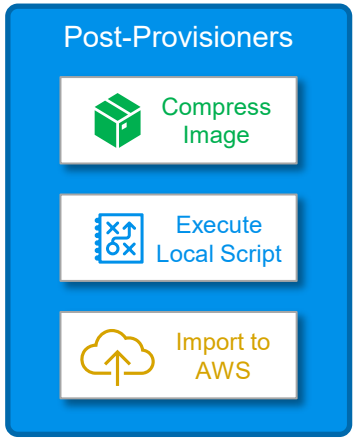
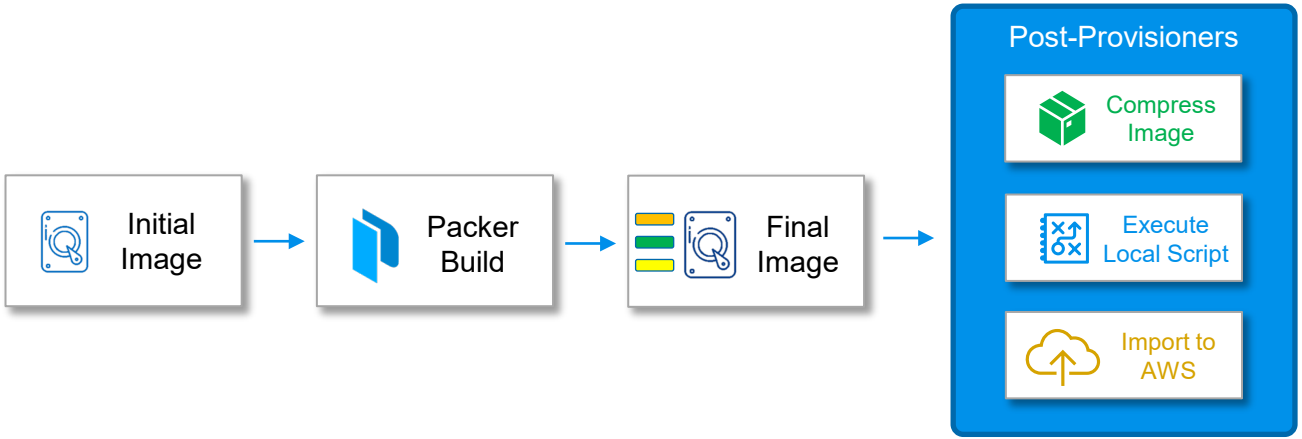
Post-Processors

Introduction to Post-Processors

- Post-processors are executed after provisioners are complete and the image is built. It can be used to upload artifacts, execute scripts, or import an image
- Post-Processors are completely optional
- Examples include:
  - Execute a local script after the build is completed (shell-local)
  - Create a machine-readable report of what was built (manifest)
  - Incorporate within a CI/CD build pipeline to be used for additional steps
  - Compute a checksum for the artifact so you can verify it later (checksum)
  - Import a package to AWS after building in your data center (AWS)
  - Convert the artifact into a Vagrant box (Vagrant)
  - Create a VMware template from the resulting build (vSphere Template)



When are Post-Processors Executed?



Using a Post-Processor

- Defined in the `build` block, each post-processor runs after each defined build. The post-processor takes the artifact from a build, uses it, and deletes the artifact after it is done (default behavior).

- Post-Processor defines a single post-processor

```
build {
  sources = [
    "source.amazon-ebs.ubuntu"
  ]
  provisioner "shell" {
    inline = [
      "echo Installing Updates",
      "sudo apt-get update",
      "sudo apt-get install -y nginx"
    ]
  }
  post-processor "manifest" {}
}
```

This Post-Processor is executed with this `build` block only

The Shell-Local Post-Processor

- The local shell post processor enables you to execute scripts locally after the machine image is built
- Very helpful for chaining tasks to your Packer build after it is completed
- You can pass in environment variables, customize how the command is executed, and specify the script to be executed

```
build {
  sources = [
    "source.amazon-ebs.ubuntu"
  ]
  provisioner "shell" {
    inline = [
      "sudo apt-get update",
    ]
  }
  post-processor "shell-local" {
    environment_vars = ["ENVIRONMENT=production"]
    scripts = ["./scripts/update_docs.sh"]
  }
}
```

The Manifest Post-Processor

- Creates a JSON file with a list all of the artifacts that Packer created during the build
- Since it's a JSON file, it's really easy to parse with `jq` and grab information from the resulting build
- The file is invoked each time a build complets and the file is updated (if it exists)
- **Default file name:** packer-manifest.json

```
build {
  sources = ["source.amazon-ebs.amazon-ebs-amazonlinux-2"]

  provisioner "file" {
    destination = "/tmp"
    source      = "files/"
  }
  post-processor "manifest" {
    output = "my-first-manifest.json"
  }
}
```

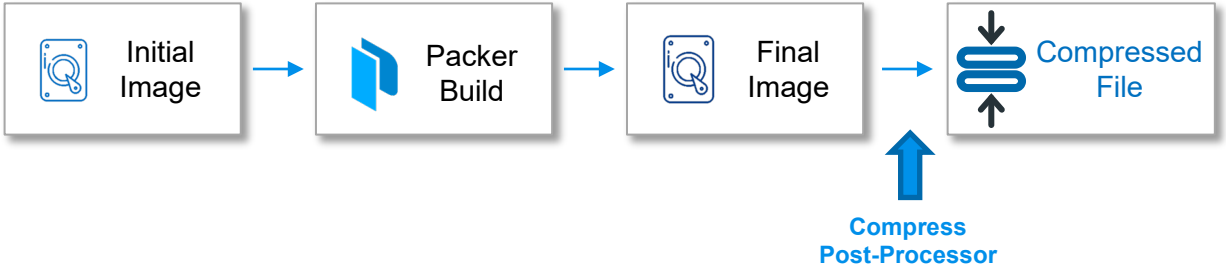
The Manifest Post-Processor

Example manifest file from a `packer build`

```
{
  "builds": [
    {
      "name": "amazon-ebs-amazonlinux-2",
      "builder_type": "amazon-ebs",
      "build_time": 1625775458,
      "files": null,
      "artifact_id": "us-east-1:ami-0f048f3536ef6836d",
      "packer_run_uuid": "e9b5fa55-d4cf-336b-5f7e-1e63e301e696",
      "custom_data": null
    }
  ],
  "last_run_uuid": "e9b5fa55-d4cf-336b-5f7e-1e63e301e696"
}
```

The Compress Post-Processor

- Takes the final artifact and compresses it into a single archive
- By default, this post-processor compresses files into a single tarball (.tar.gz file)
- However, the following extensions are supported: .zip, .gz, .tar.gz, .lz4, and .tar.lz4
- Very helpful if you're building packages locally – vSphere, Vagrant, etc.



The Compress Post-Processor

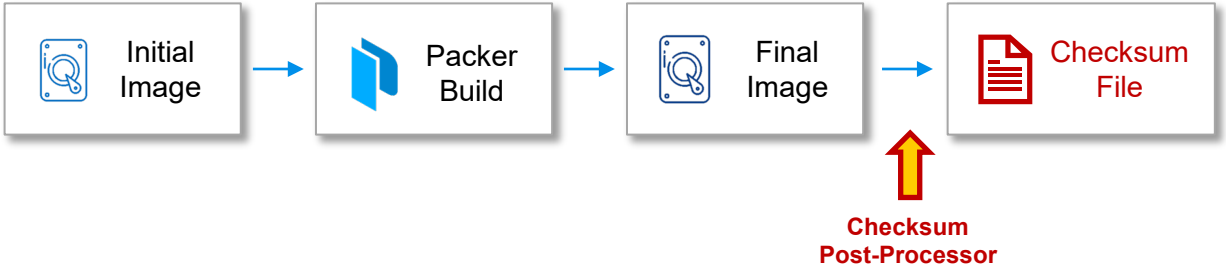
Example of a compress Post-Processor

```
build {
  sources = ["source.amazon-ebs.amazonlinux-2"]

  post-processor "compress" {
    output = "{{.BuildName}}-image.zip"
  }
}
```

The Checksum Post-Processor

- Computes a checksum for the current artifact
- Useful to validate no changes occurred to the artifact since running the `packer build`
- Can be used during validation phase of a CI/CD pipeline



The Checksum Post-Processor

Example of a checksum Post-Processor

```
build {
  sources = ["source.amazon-ebs.amazonlinux-2"]

  post-processor "checksum" {
      checksum_types = ["sha1", "sha256"]
      output = "packer_{{.BuildName}}_{{.ChecksumType}}.checksum"
  }
```

END OF SECTION