Writing
Packer Templates

Packer Templates

- The core functionality and behavior of HashiCorp Packer is defined by a **template**
- Templates consist of declarations and command, such as what plugins (builders, provisioners, etc.) to use, how to configure the plugins, and what order to run them

Packer templates support two formats

HCL2
(HashiCorp Configuration Language)

{ JSON }

This course focuses on writing Packer templates using the HCL2 format

Packer Templates

The Future of Packer is Now

HashiCorp is moving to **HCL2** as the standard format for Packer 1.7+

JSON was previously the standard format

Mighty Fine Design

HCL is designed to strike a balance between human-readable and machine-parsable

Packer templates are very easy to develop and read

Look Familiar?

If you have Terraform experience, Packer templates are going to look very familiar

HCL is the same configuration language as Terraform and other HashiCorp products

Note: JSON templates will enter a legacy support phase before being deprecated

HCL Formatting

- Configuration format is VCS friendly
  (multi-line lists, trailing commas, auto-formatting)
- Only code blocks built into the HCL language are available for use
- Packer uses a standard file name for simplicity <name>pkr.hcl
- Uses Syntax Constructs like Blocks and Arguments
- New features will only be implemented for the HCL format moving forward

HCL Syntax

syntax.pkr.hcl

```
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  <IDENTIFIER> = <EXPRESSION>
  <IDENTIFIER> = <EXPRESSION>
  <IDENTIFIER> = <EXPRESSION>
}
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  <IDENTIFIER> = <EXPRESSION>
  <IDENTIFIER> = <EXPRESSION>
  <IDENTIFIER> = <EXPRESSION>
}
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  <IDENTIFIER> = <EXPRESSION>
  <IDENTIFIER> = <EXPRESSION>
  <IDENTIFIER> = <EXPRESSION>
}
```

Block

Arguments

Arguments

HCL File Structure

```
.
├── README.md
├── ubuntu.pkr.hcl
├── centos.pkr.hcl
├── rhel.pkr.hcl
├── web-app.pkr.hcl
├── prod.pkrvars.hcl
```

Build all templates in a directory

Terminal

```
$ packer build /opt/packer/templates
```

Build using a single template

Terminal

```
$ packer build rhel.pkr.hcl
```

HCL Example

```
source "amazon-ebs" "aws-example" {
  ami_name      = "${var.ami_name}"
  instance_type = "t3.medium"
  region        = "us-east-1"
  source_ami_filter {
    filters = {
      name                = "${var.source_ami_name}"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    owners = ["amazon"]
  }
  ssh_username = "ec2-user"
  subnet_id    = "${var.subnet_id}"
  tags = {
    Name = "${var.ami_name}"
  }
  vpc_id = "vpc-1234567890"
}
```

```
build {
  sources = ["source.amazon-ebs.autogenerated_1"]

  provisioner "file" {
    destination = "/tmp"
    source      = "files"
  }
  provisioner "shell" {
    script = "scripts/setup.sh"
  }
  provisioner "shell" {
    script = "scripts/vault.sh"
  }
}
```

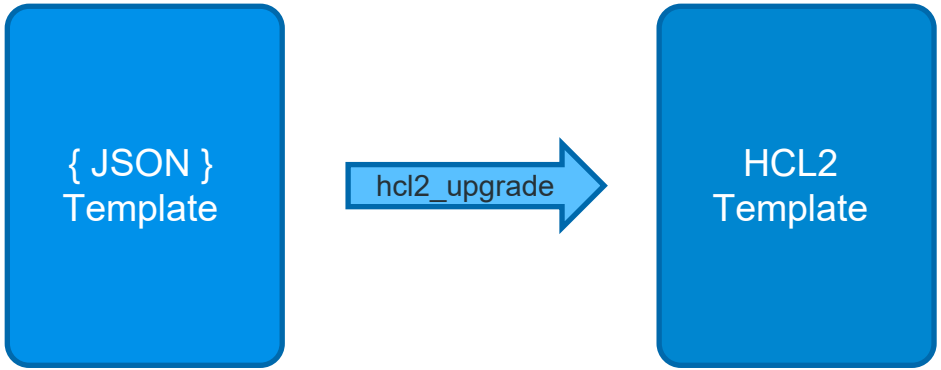
JSON Example

```
{
  "variables": {
    "ami_prefix": "amzn2",
    "ami_name": "{{user `ami_prefix`}}-{{user `app_name`}}-ent-{{user `consul_version`}}-{{timestamp}}",
    "region": "us-east-1",
    "vpc_id": "vpc-1234567890",
    "subnet_id": "subnet-1234567890",
    "ssh_username": "ec2-user",
    "source_ami_name": "amzn2-ami-hvm-2.0.20200304.0-x86_64-gp2",
    "source_ami_owner": "amazon"
  },
  "builders": [{
    "type": "amazon-ebs",
    "region": "{{user `region`}}",
    "vpc_id": "{{ user `vpc_id`}}",
    "subnet_id": "{{ user `subnet_id`}}",
    "source_ami_filter": {
      "filters": {
        "virtualization-type": "hvm",
        "name": "{{user `source_ami_name`}}",
        "root-device-type": "ebs"
      },
      "owners": ["{{user `source_ami_owner`}}"]
    },
    "tags": {
      "Name": "{{user `ami_name`}}"
    }
  }],
  "provisioners": [
    {
      "type": "file",
      "source": "files",
      "destination": "/tmp"
    },
```

Converting Old Templates to HCL

- If you have older, JSON formatted templates, you can easily convert them using the `packer hcl2_upgrade` command

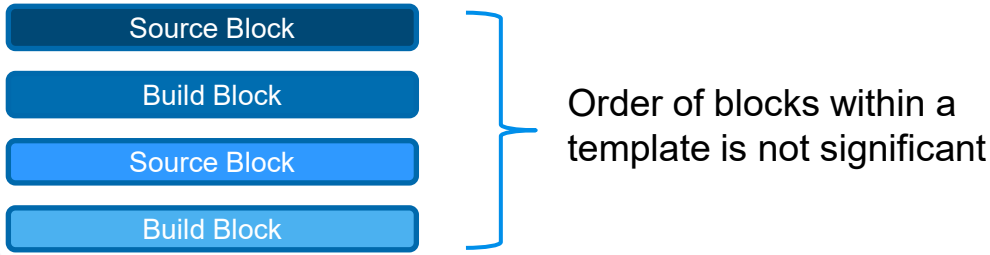
- All fields of builders, provisioners and post-processors have a 1:1 correspondence with a few exceptions.



HCL Syntax

Block Organization

- In general, the ordering of root blocks is not significant within a Packer template since Packer uses a declarative model. References to other resources do not depend on the order they are defined.
- Blocks can even span multiple Packer template files.
- The order of provisioner or post-processor blocks within a build is the only major feature where block order matters.



Commenting in HCL2

HCL2

- HCL2 supports comment to use throughout your configuration file
  - # - single line comment
  - // - single line comment
  - /* and */ - start and stop delimiters – might span over multiple lines

Terminal

```
# this is a comment
source "amazon-ebs" "example" {
  ami_name = "abc123"
}
// this is also a comment
```

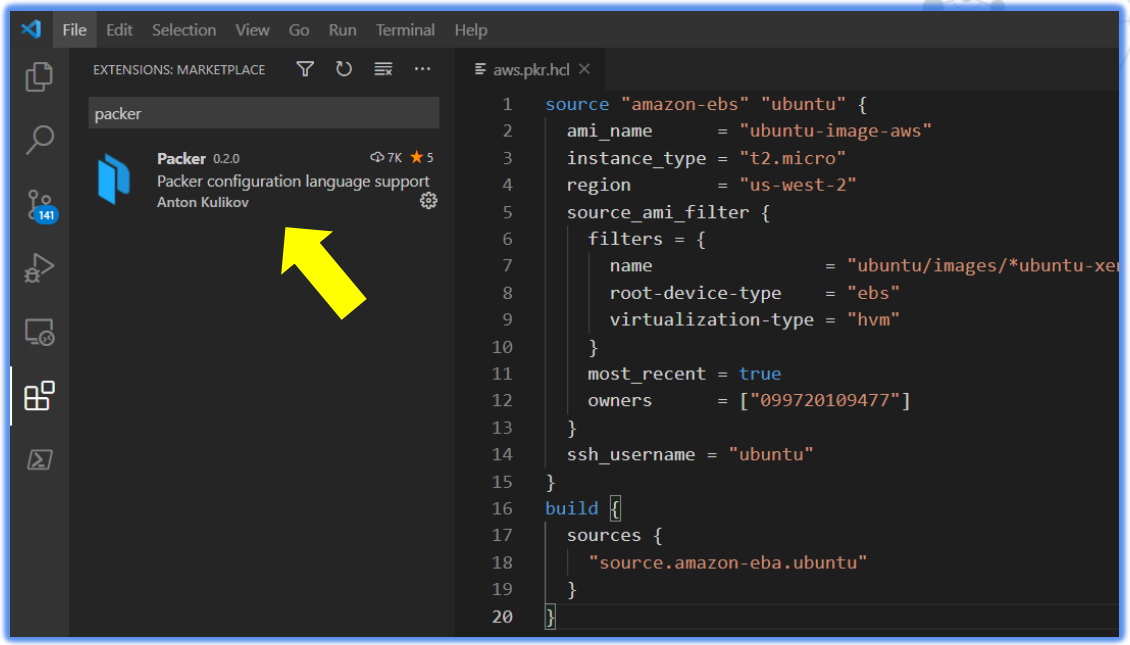
Terminal

```
/* <-this is a multi-line comment
source "amazon-ebs" "example" {
  ami_name = "abc123"
}
*/
variable "example" {
```

Everything in green is commented out here

Syntax Highlighting

- Plugins for HCL exist for most major editors.
- VS Code Extension for Packer adds syntax support for the Packer HCL configuration language.



Interpolation Syntax

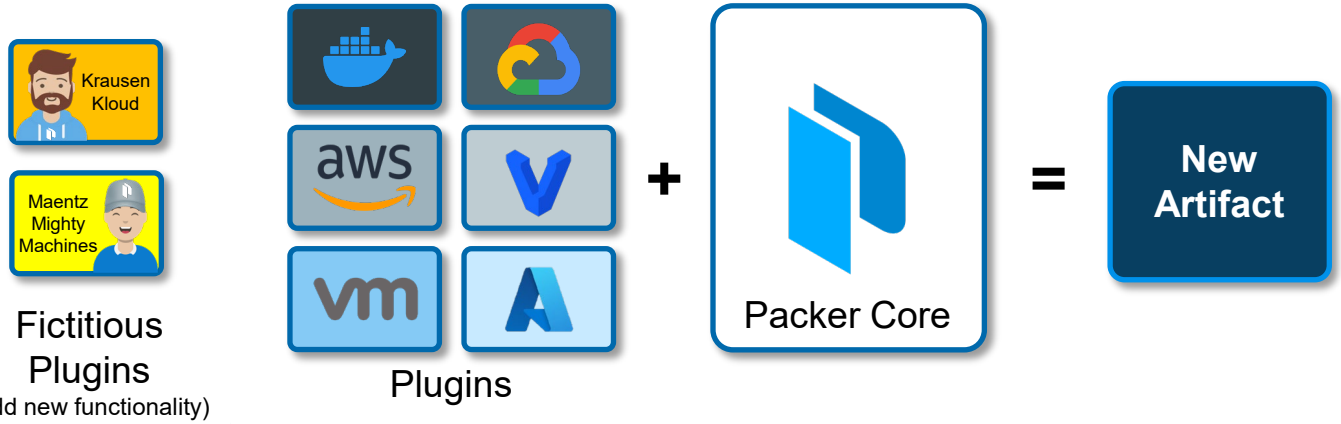
```
source "amazon-ebs" "amazon-ebs-amazonlinux-2" {
  ami_description             = "Vault - Amazon Linux 2"
  ami_name                    = "vault-amazonlinux2"
  ami_regions                 = ["us-east-1"]
  ami_virtualization_type     = "hvm"
  associate_public_ip_address = true
  force_delete_snapshot       = true
  force_deregister            = true
  instance_type               = "m5.large"
  region                      = var.aws_region
  source_ami                  = data.amazon-ami.amazon-linux-2.id
  tags = {
    Name           = "HashiCorp Vault"
    OS             = "Amazon Linux 2"
  }
  subnet_id                   = var.subnet_id
  vpc_id                      = var.vpc_id
}

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

  provisioner "file" {
    destination = "/tmp/vault.zip"
    source      = var.vault_zip
  }
}
```

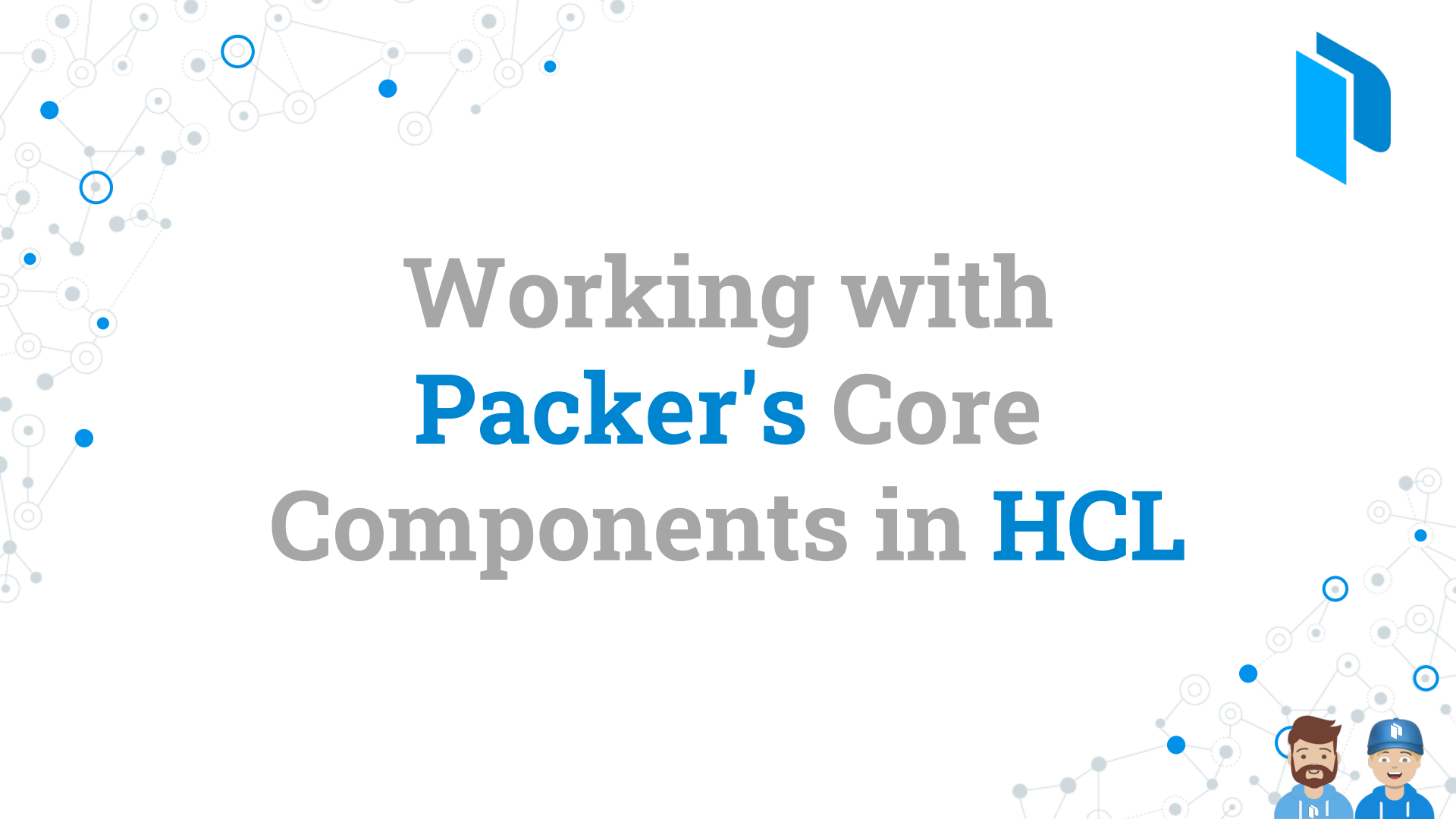
- Like Terraform, we can use interpolation syntax to refer to other blocks within the template
- Allows us to organize code as well as reuse values that are already defined or have been retrieved

Plugin Architecture

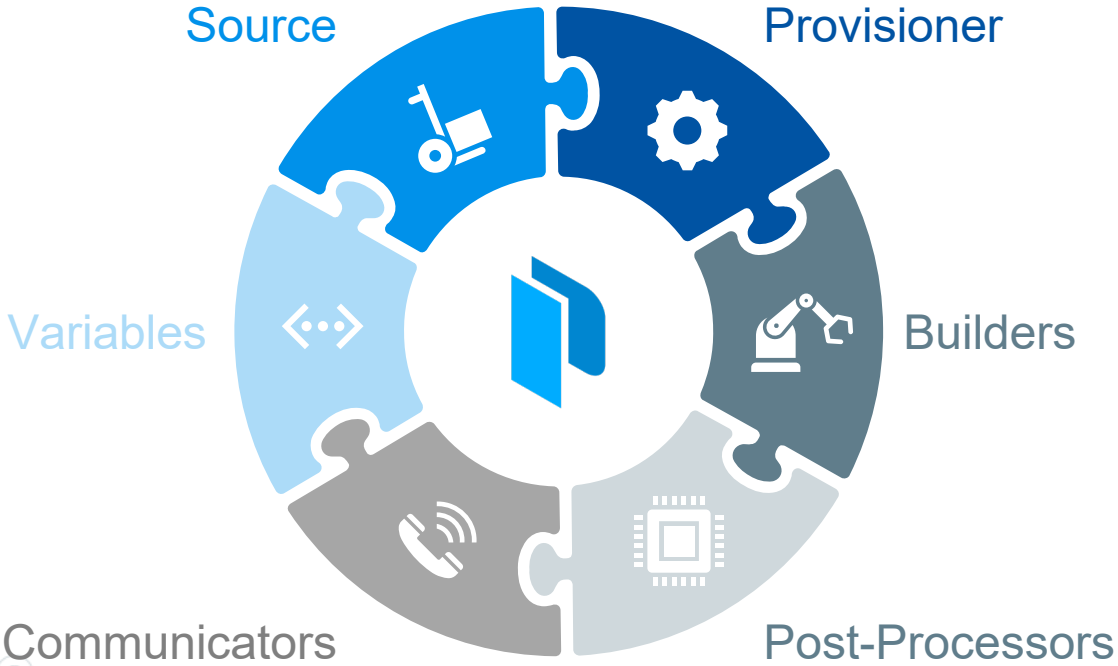
- Builders, provisioners, post-processors, and data sources are simply plugins that are consumed during the Packer build process

- This allows new functionality to be added to Packer without modifying the core source code




Working with
Packer's Core
Components in HCL

Core Components



Source Blocks

Only available in HCL templates

Source blocks define the initial image to use to create your customized image, how to launch the image, and how to connect to the image.

```
source "amazon-ebs" "ubuntu" {
  instance_type = "t2.micro"
  region        = "us-west-2"
  ami_id        = "ami-a1b2c3d4e5"
}
```

Sources can be used across multiple builds and references in builder blocks

Only available in HCL templates

Source Blocks

How to Launch Image & What Virtualization to Use

Source blocks define what kind of virtualization to use for the image, how to launch the image

In the example, the `amazon-ebs` builder configuration launches a t2.micro AMI in the us-west-2 region

amazon.pkr.hcl

```
source "amazon-ebs" "ubuntu" {
  ami_name      = "ubuntu-image-aws"
  instance_type = "t2.micro"
  region        = "us-west-2"
  source_ami_filter {
    filters = {
      name                = "ubuntu/images/*ubuntu-xenial-16.04-amd-server-*"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
}
```

Only available in HCL templates

Source Blocks

How to Connect to Image

Source blocks defining how to connect to the image.

In the example, the SSH communicator is used to allow Packer to SSH into the EC2 instance.

amazon.pkr.hcl

```
source "amazon-ebs" "ubuntu" {
  ami_name      = "ubuntu-image-aws"
  instance_type = "t2.micro"
  region        = "us-west-2"
  source_ami_filter {
    filters = {
      name                = "ubuntu/images/*ubuntu-xenial-16.04-amd-server-*"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
  ssh_username = "ubuntu"
}
```

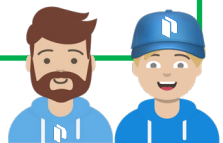
Builders

Build blocks are used in tandem with source blocks and define what Packer should do with the image after it is launched.

The example template will be used to build an AWS Ubuntu AMI in the us-west-2 region. It references the AMI defined in the source block.

amazon.pkr.hcl

```
source "amazon-ebs" "ubuntu" {
  ami_name      = "ubuntu-image-aws"
  instance_type = "t2.micro"
  region        = "us-west-2"
  source_ami_filter {
    filters = {
      name                = "ubuntu/images/*ubuntu-xenial-16.04-amd-server-*"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"]
  }
  ssh_username = "ubuntu"
}
build {
  sources {
    "source.amazon-ebs.ubuntu"
  }
}
```



Provisioners

Provisioners install and configure the machine image after it reboots. You would use one or many provisioners to customize the image as needed. Part of the build block.

File Provisioner

Shell Provisioners

gcp.pkr.hcl

```
build {
  sources = ["sources.googlecompute.debian-build"]

  provisioner "file" {
    destination = "/tmp"
    source      = "files"
  }
  provisioner "shell" {
    script = "scripts/setup.sh"
  }
  provisioner "shell" {
    inline = ["echo ${var.deployment_version} > ~/DEPLOYMENT_VERSION"]
  }
}
```

1st

2nd

3rd

Post-Processors

Post-processors can be used to specify what to do after the image is created. Part of the build block and is not mandatory.

Post Processor

aws.pkr.hcl

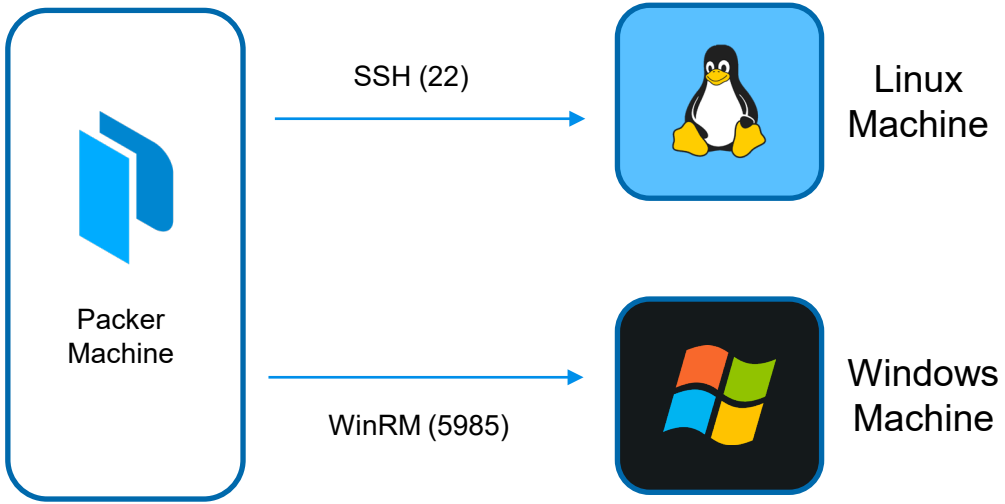
```
build {
  sources {
    "source.amazon-ebs.ubuntu"
  }
  post-processor "shell-local" {
    inline = ["rm /tmp/script.sh"]
  }
}
```

Communicators

Communicators are the mechanism that Packer uses to upload files and/or execute scripts when creating the final artifact.

Communicators are configured within the Source block of the Packer template.

- SSH - default
- WinRM
- Builder-Specific (docker exec)



Communicators

Linux

aws-centos.pkr.hcl

```
source "amazon-ebs" "centos" {
  ami_name      = "packer-centos-aws-{{timestamp}}"
  instance_type = "t2.micro"
  region        = "us-west-2"
  ami_regions   = ["us-west-2"]
  source_ami_filter {
    filters = {
      name                = "CentOS Linux 7 x86_64 HVM EBS *"
      product-code        = "aw0evgkw8e5c1q413zgy5pjce"
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["679593333241"]
  }
  ssh_username = "centos"
}
```

SSH is the default Communicator so we don't need to specify the communicator type here

Communicators

Windows

Communicator

Communicator Configuration

aws-windows.pkr.hcl

```
source "amazon-ebs" "windows-2012r2" {
  ami_name       = "my-windows-2012-aws-{{timestamp}}"
  communicator   = "winrm"
  instance_type  = "t2.micro"
  region         = "us-east-1"
  source_ami     = "${data.amazon-ami.windows_2012r2.id}"
  user_data_file = "./scripts/SetUpWinRM.ps1"
  winrm_insecure = true
  winrm_use_ssl  = true
  winrm_username = "Administrator"
  tags = {
    "Name"        = "MyWindowsImage"
    "Environment" = "Production"
    "OS_Version"  = "Windows"
    "Release"     = "Latest"
    "Created-by"  = "Packer"
  }
}
```

Remember that Windows is not the default Communicator so it must be defined

Variables

You can use input variables (*aka variables*) or local variables (*aka locals*) to define values for arguments throughout a Packer template.

variable block

variable block

aws.pkr.hcl

```
variable "subnet_id" {
  type    = string
  default = "subnet-1a2b3c4d5e"
}
variable "region" {
  type    = string
  default = "us-east-1"
}

source "amazon-ebs" "amazon-linux2" {
  ami_name      = local.ami_name
  instance_type = "t3.medium"
  region        = var.region
  source_ami    = data.amazon-ami.amazon-linux.id
  ssh_username  = var.ssh_username
  subnet_id     = var.subnet_id
  tags = {
    Name = local.ami_name
  }
  vpc_id = var.vpc_id
}
```

Defined in Variable Block

Referenced in Source Block

END OF SECTION