Joys of Packer:

same code, multiple clouds

Who am I?

Jordi Gutiérrez Hermoso

Who am I?

Jordi Gutiérrez Hermoso

Grist

Email: jordi@getgrist.com
Fediverse: @jordigh@mathstodon.xzy

Who am I?

Jordi Gutiérrez Hermoso

mercurial

Grist

Email: jordi@getgrist.com
Fediverse: @jordigh@mathstodon.xzy

Who am I?

Jordi Gutiérrez Hermoso

- NES enthusiast (let’s talk 6502)
- Mercurial contributor (still using it!)
- Grist systems developer

Nes Dev

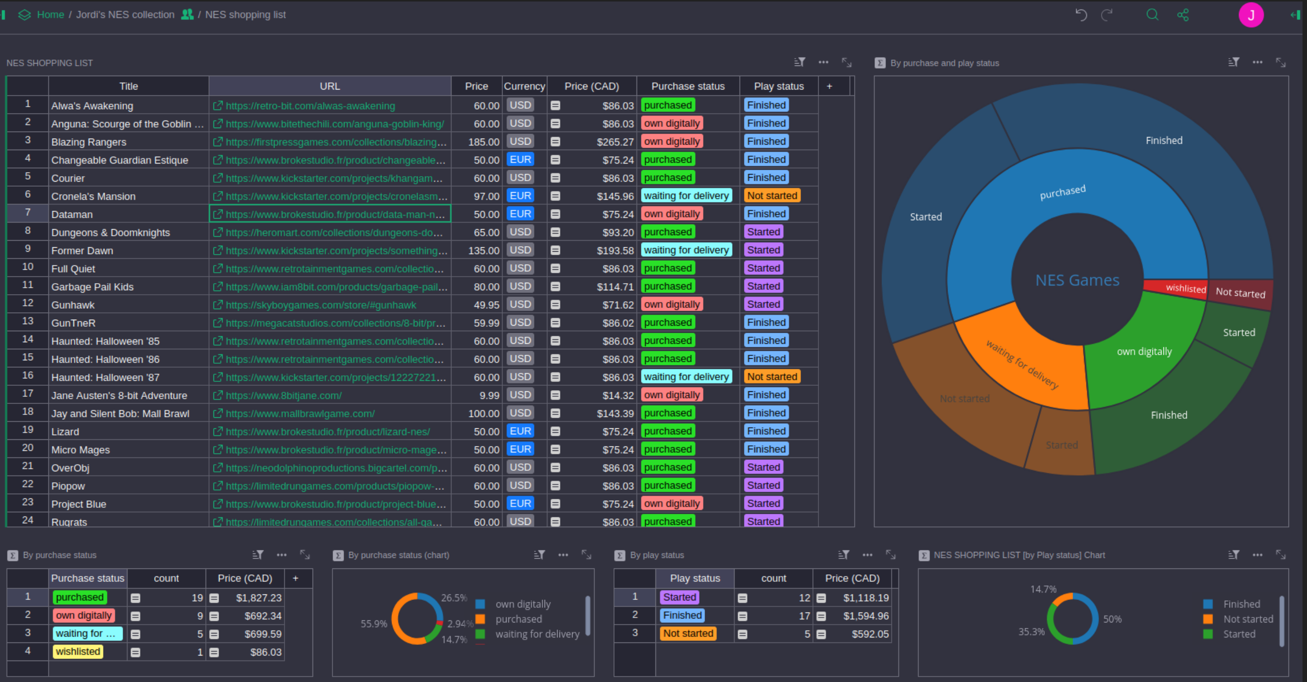
mercurial

Grist

Email: jordi@getgrist.com
Fediverse: @jordigh@mathstodon.xzy

Grist for

tracking NES purchases



Home / Jordi's NES collection / NES shopping list

NES SHOPPING LIST

	Title	URL	Price	Currency	Price (CAD)	Purchase status	Play status	+
1	Alwa's Awakening	https://retro-bit.com/alwas-awakening	60.00	USD	$86.03	purchased	Finished	
2	Anguna: Scourge of the Goblin …	https://www.bitethechili.com/anguna-goblin-king/	60.00	USD	$86.03	own digitally	Finished	
3	Blazing Rangers	https://firstpressgames.com/collections/blazing…	185.00	USD	$265.27	own digitally	Finished	
4	Changeable Guardian Estique	https://www.brokestudio.fr/product/changeable…	50.00	EUR	$75.24	purchased	Finished	
5	Courier	https://www.kickstarter.com/projects/khangam…	60.00	USD	$86.03	purchased	Finished	
6	Cronela's Mansion	https://www.kickstarter.com/projects/cronelasm…	97.00	EUR	$145.96	waiting for delivery	Not started	
7	Dataman	https://www.brokestudio.fr/product/data-man-n…	50.00	EUR	$75.24	own digitally	Finished	
8	Dungeons & Doomknights	https://heromart.com/collections/dungeons-do…	65.00	USD	$93.20	purchased	Started	
9	Former Dawn	https://www.kickstarter.com/projects/something…	135.00	USD	$193.58	waiting for delivery	Started	
10	Full Quiet	https://www.retrotainmentgames.com/collectio…	60.00	USD	$86.03	purchased	Started	
11	Garbage Pail Kids	https://www.iam8bit.com/products/garbage-pail…	80.00	USD	$114.71	purchased	Started	
12	Gunhawk	https://skyboygames.com/store/#gunhawk	49.95	USD	$71.62	own digitally	Started	
13	GunTneR	https://megacatstudios.com/collections/8-bit/pr…	59.99	USD	$86.02	purchased	Finished	
14	Haunted: Halloween '85	https://www.retrotainmentgames.com/collectio…	60.00	USD	$86.03	purchased	Finished	
15	Haunted: Halloween '86	https://www.retrotainmentgames.com/collectio…	60.00	USD	$86.03	purchased	Finished	
16	Haunted: Halloween '87	https://www.kickstarter.com/projects/12227221…	60.00	USD	$86.03	waiting for delivery	Not started	
17	Jane Austen's 8-bit Adventure	https://www.8bitjane.com/	9.99	USD	$14.32	own digitally	Finished	
18	Jay and Silent Bob: Mall Brawl	https://www.mallbrawlgame.com/	100.00	USD	$143.39	purchased	Finished	
19	Lizard	https://www.brokestudio.fr/product/lizard-nes/	50.00	EUR	$75.24	purchased	Finished	
20	Micro Mages	https://www.brokestudio.fr/product/micro-mage…	50.00	EUR	$75.24	purchased	Finished	
21	OverObj	https://neodolphinoproductions.bigcartel.com/p…	60.00	USD	$86.03	purchased	Started	
22	Piopow	https://limitedrungames.com/products/piopow-…	60.00	USD	$86.03	purchased	Started	
23	Project Blue	https://www.brokestudio.fr/product/project-blue…	50.00	EUR	$75.24	own digitally	Started	
24	Rugrats	https://limitedrungames.com/collections/all-ga…	60.00	USD	$86.03	purchased	Started	

By purchase and play status

By purchase status

	Purchase status	count	Price (CAD)	+
1	purchased	19	$1,827.23	
2	own digitally	9	$692.34	
3	waiting for …	5	$699.59	
4	wishlisted	1	$86.03	

By purchase status (chart)

By play status

	Play status	count	Price (CAD)
1	Started	12	$1,118.19
2	Finished	17	$1,594.96
3	Not started	5	$592.05

NES SHOPPING LIST [by Play status] Chart

What is Packer?

A tool for building virtual machines *images*

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)
  - AWS

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)
  - AWS
  - Azure

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)
  - AWS
  - Azure
  - Digital Ocean

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)
  - AWS
  - Azure
  - Digital Ocean
  - VMware

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)
  - AWS
  - Azure
  - Digital Ocean
  - VMware
  - Vagrant

What is Packer?

A tool for building virtual machines *images*

- Many different VM targets (via plugins)
  - AWS
  - Azure
  - Digital Ocean
  - VMware
  - Vagrant
  - Virtualbox

What is Packer?

Built by Hashicorp

What is Packer?

Built by Hashicorp

- Uses HCL, a declarative language (was: JSON)

What is Packer?

Built by Hashicorp

- Uses HCL, a declarative language (was: JSON)
- Lots of community-maintained add-ons

What is Packer?

Built by Hashicorp

- Uses HCL, a declarative language (was: JSON)
- Lots of community-maintained add-ons
- Does not replace Chef/Ansible/Puppet

Why Packer?

The main use case (well, our use case anyway)

Why Packer?

The main use case (well, our use case anyway)

- We want to sell our software on cloud providers
  - AWS
  - Azure
  - Maybe Digital Ocean
  - Maybe others later

Why Packer?

The main use case (well, our use case anyway)

- We want to sell our software on cloud providers
  - AWS
  - Azure
  - Maybe Digital Ocean
  - Maybe others later
- The easiest way is machine images

Why Packer?

The main use case (well, our use case anyway)

- We want to sell our software on cloud providers
  - AWS
  - Azure
  - Maybe Digital Ocean
  - Maybe others later
- The easiest way is machine images
- Thus, we want a unified way to build them

The `grist.pkr.hcl` file

Let’s look at some HCL code

https://github.com/gristlabs/grist-pack/blob/main/grist.pkr.hcl

The `grist.pkr.hcl` file

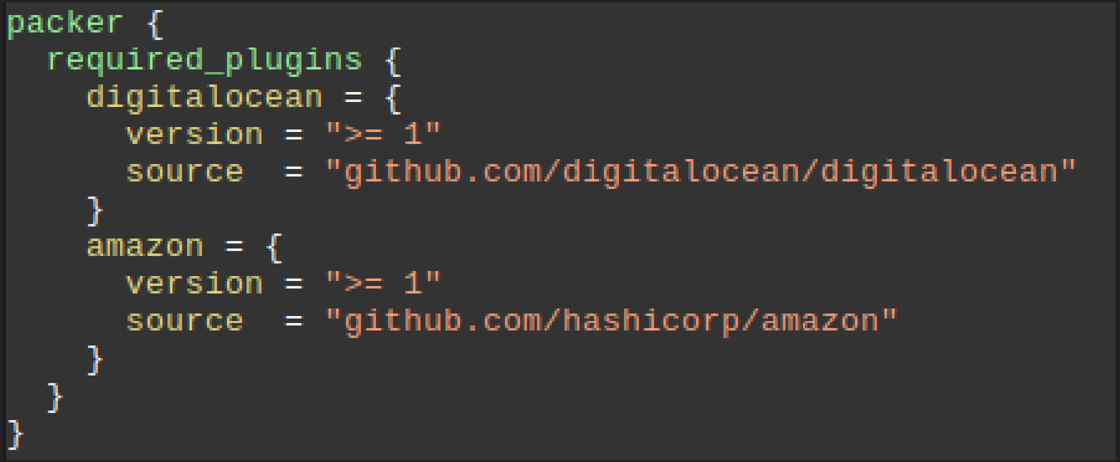
Let’s look at some HCL code

If you want the punchline or to follow along...

https://github.com/gristlabs/grist-pack/blob/main/grist.pkr.hcl

The `grist.pkr.hcl` file

Let’s look at some HCL code

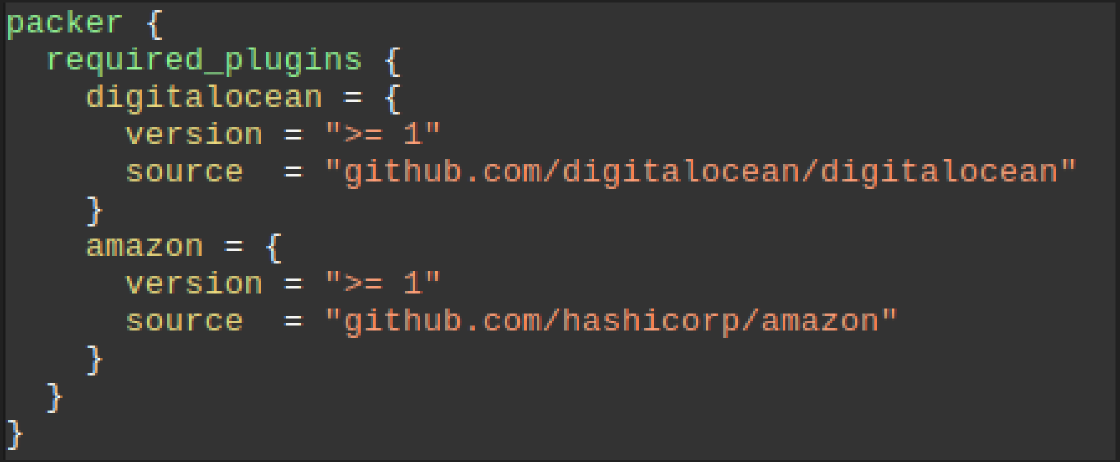


```
packer {
  required_plugins {
    digitalocean = {
      version = ">= 1"
      source  = "github.com/digitalocean/digitalocean"
    }
    amazon = {
      version = ">= 1"
      source  = "github.com/hashicorp/amazon"
    }
  }
}
```

The `grist.pkr.hcl` file

Let’s look at some HCL code

Plugins define the types of images we’ll build

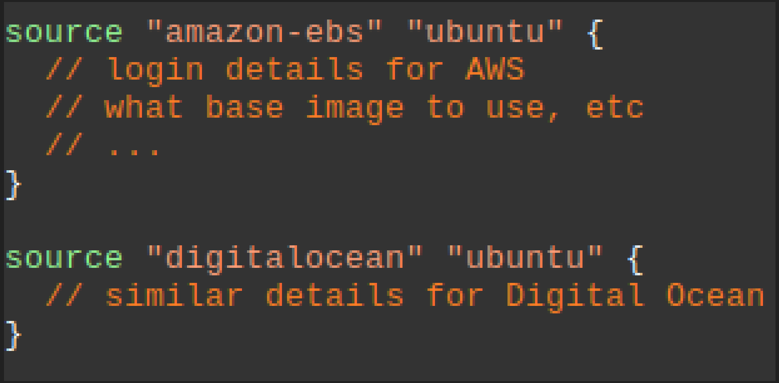


```
packer {
  required_plugins {
    digitalocean = {
      version = ">= 1"
      source  = "github.com/digitalocean/digitalocean"
    }
    amazon = {
      version = ">= 1"
      source  = "github.com/hashicorp/amazon"
    }
  }
}
```

The `grist.pkr.hcl` file

Let’s look at some HCL code

Sources define what Packer will build

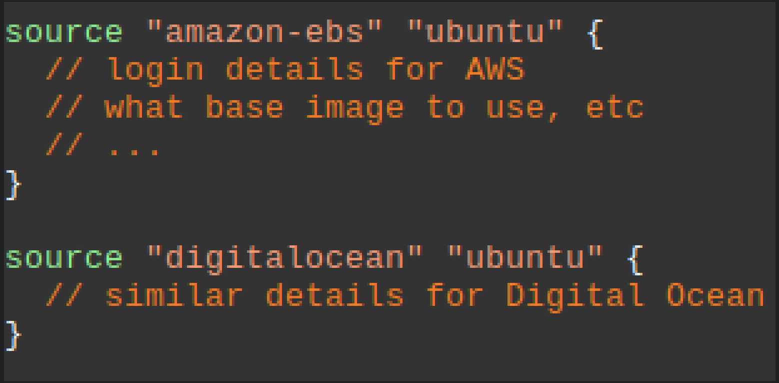


```
source "amazon-ebs" "ubuntu" {
  // login details for AWS
  // what base image to use, etc
  // ...
}

source "digitalocean" "ubuntu" {
  // similar details for Digital Ocean
}
```

The grist.pkr.hcl file

Let’s look at some HCL code

Sources define what Packer will build



```
source "amazon-ebs" "ubuntu" {
  // login details for AWS
  // what base image to use, etc
  // ...
}

source "digitalocean" "ubuntu" {
  // similar details for Digital Ocean
}
```

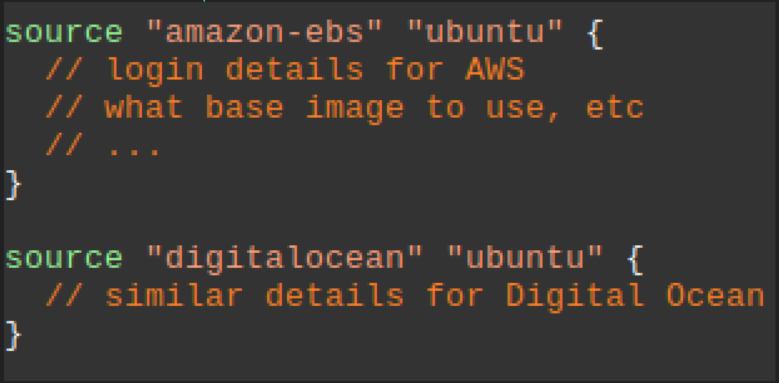
(they look like targets, but these blocks define sources for builders)

The `grist.pkr.hcl` file

Let’s look at some HCL code

Sources define what Packer will build

Builder type (defined by AWS plugin)



```
source "amazon-ebs" "ubuntu" {
  // login details for AWS
  // what base image to use, etc
  // ...
}

source "digitalocean" "ubuntu" {
  // similar details for Digital Ocean
}
```

(they look like targets, but these blocks define sources for builders)

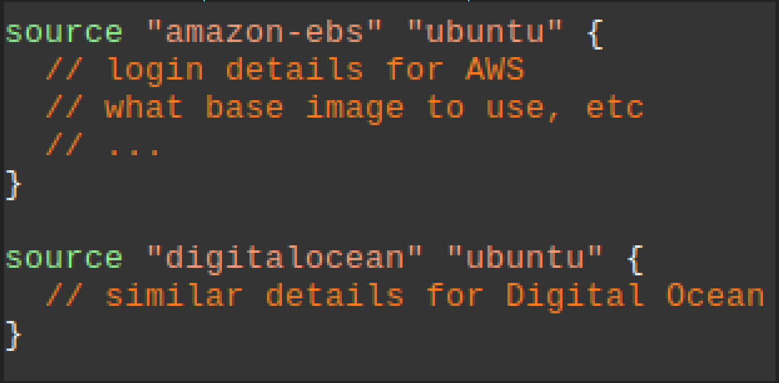
The `grist.pkr.hcl` file

Let’s look at some HCL code

Sources define what Packer will build

Builder type (defined by AWS plugin)

Identifier of builder (generic name)

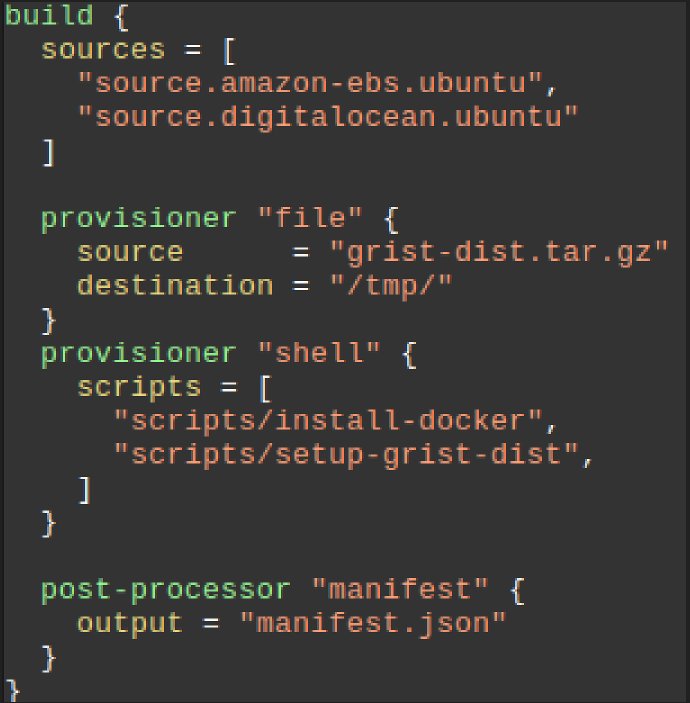


```
source "amazon-ebs" "ubuntu" {
  // login details for AWS
  // what base image to use, etc
  // ...
}

source "digitalocean" "ubuntu" {
  // similar details for Digital Ocean
}
```

(they look like targets, but these blocks define sources for builders)

The `grist.pkr.hcl` file

Let’s look at some HCL code

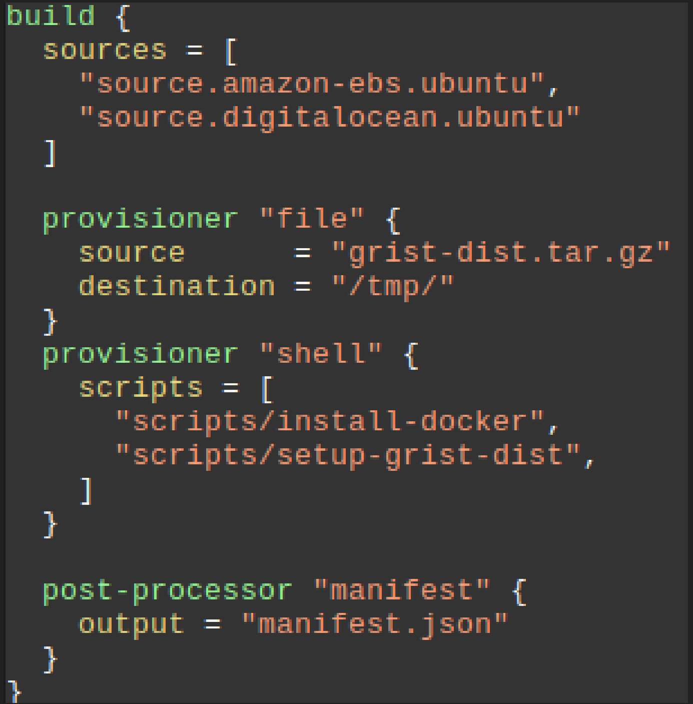


```
build {
  sources = [
    "source.amazon-ebs.ubuntu",
    "source.digitalocean.ubuntu"
  ]

  provisioner "file" {
    source      = "grist-dist.tar.gz"
    destination = "/tmp/"
  }
  provisioner "shell" {
    scripts = [
      "scripts/install-docker",
      "scripts/setup-grist-dist",
    ]
  }

  post-processor "manifest" {
    output = "manifest.json"
  }
}
```

The grist.pkr.hcl file

Let’s look at some HCL code

The build section uses the defined sources



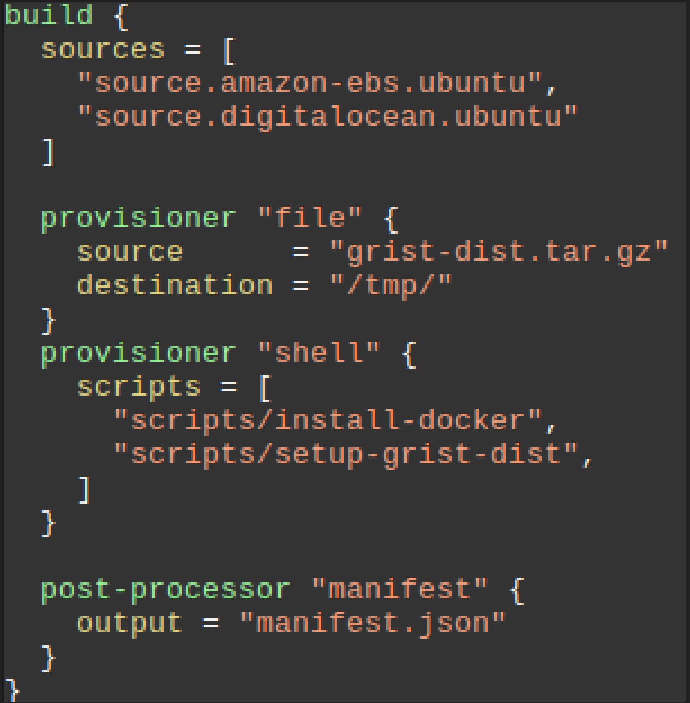
```
build {
  sources = [
    "source.amazon-ebs.ubuntu",
    "source.digitalocean.ubuntu"
  ]

  provisioner "file" {
    source      = "grist-dist.tar.gz"
    destination = "/tmp/"
  }
  provisioner "shell" {
    scripts = [
      "scripts/install-docker",
      "scripts/setup-grist-dist",
    ]
  }

  post-processor "manifest" {
    output = "manifest.json"
  }
}
```

The `grist.pkr.hcl` file

Let’s look at some HCL code

The build section uses the defined sources

Provisioners define actions to take



```
build {
  sources = [
    "source.amazon-ebs.ubuntu",
    "source.digitalocean.ubuntu"
  ]

  provisioner "file" {
    source      = "grist-dist.tar.gz"
    destination = "/tmp/"
  }
  provisioner "shell" {
    scripts = [
      "scripts/install-docker",
      "scripts/setup-grist-dist",
    ]
  }

  post-processor "manifest" {
    output = "manifest.json"
  }
}
```

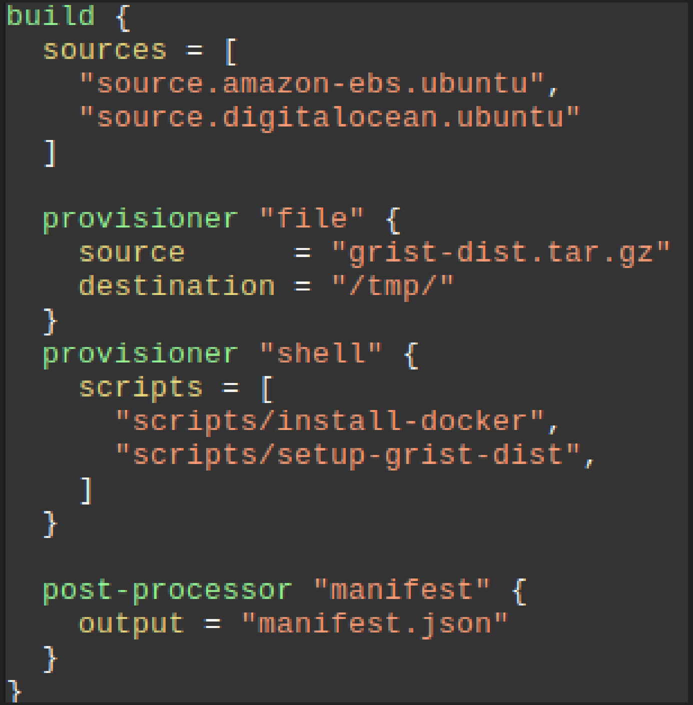
The `grist.pkr.hcl` file

Let’s look at some HCL code

The build section uses the defined sources

Provisioners define actions to take

Processors define ancilliary steps

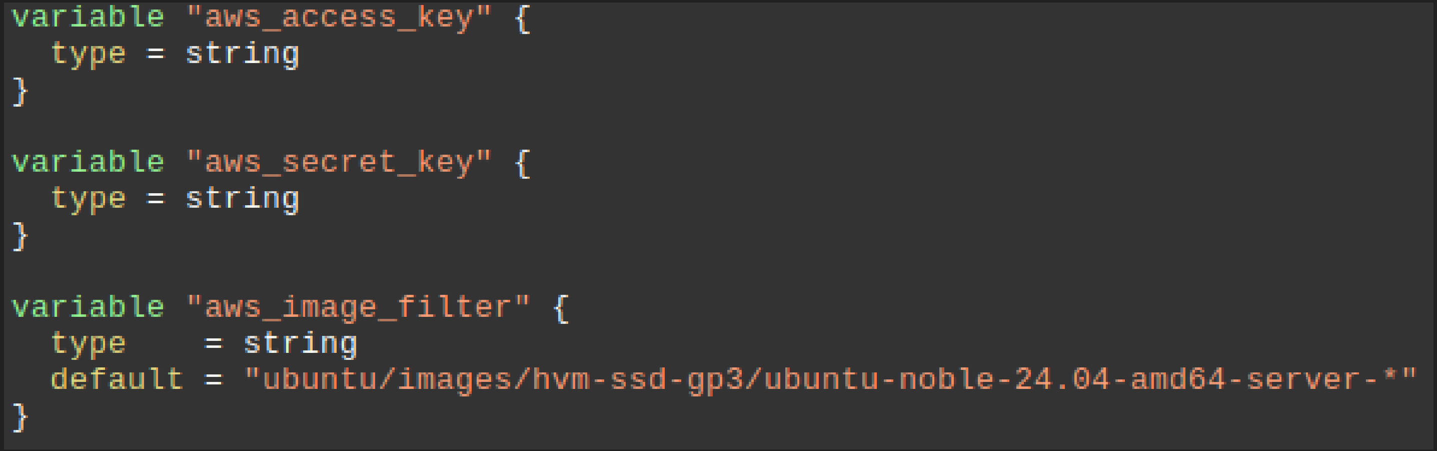


```
build {
  sources = [
    "source.amazon-ebs.ubuntu",
    "source.digitalocean.ubuntu"
  ]

  provisioner "file" {
    source      = "grist-dist.tar.gz"
    destination = "/tmp/"
  }
  provisioner "shell" {
    scripts = [
      "scripts/install-docker",
      "scripts/setup-grist-dist",
    ]
  }

  post-processor "manifest" {
    output = "manifest.json"
  }
}
```

The `grist.pkr.hcl` file

More details on configuring sources

- Define some *static* variables

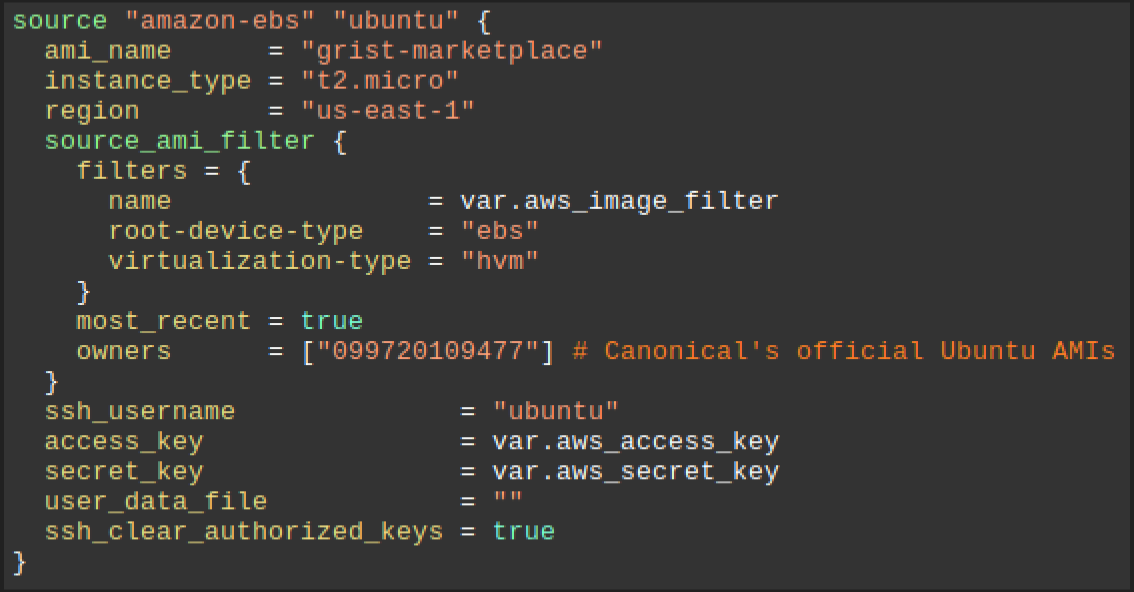


```
variable "aws_access_key" {
  type = string
}

variable "aws_secret_key" {
  type = string
}

variable "aws_image_filter" {
  type    = string
  default = "ubuntu/images/hvm-ssd-gp3/ubuntu-noble-24.04-amd64-server-*"
}
```

The grist.pkr.hcl file

More details on configuring sources

- Use those variables to define the full source block



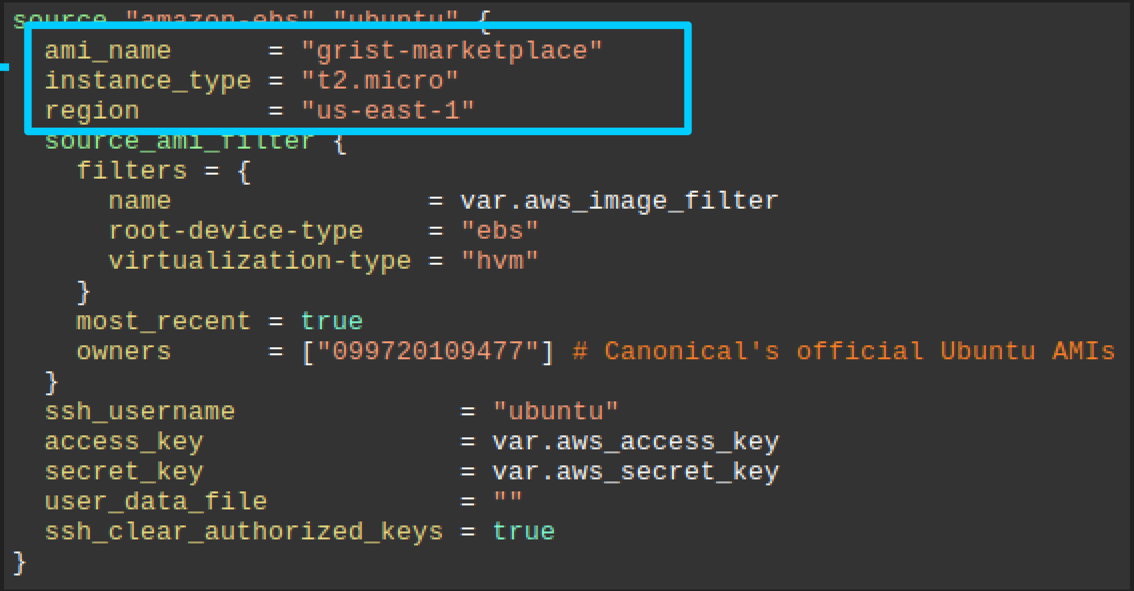
```
source "amazon-ebs" "ubuntu" {
  ami_name      = "grist-marketplace"
  instance_type = "t2.micro"
  region        = "us-east-1"
  source_ami_filter {
    filters = {
      name                = var.aws_image_filter
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"] # Canonical's official Ubuntu AMIs
  }
  ssh_username                = "ubuntu"
  access_key                  = var.aws_access_key
  secret_key                  = var.aws_secret_key
  user_data_file              = ""
  ssh_clear_authorized_keys = true
}
```

The grist.pkr.hcl file

More details on configuring sources

- Use those variables to define the full source block

AMI details



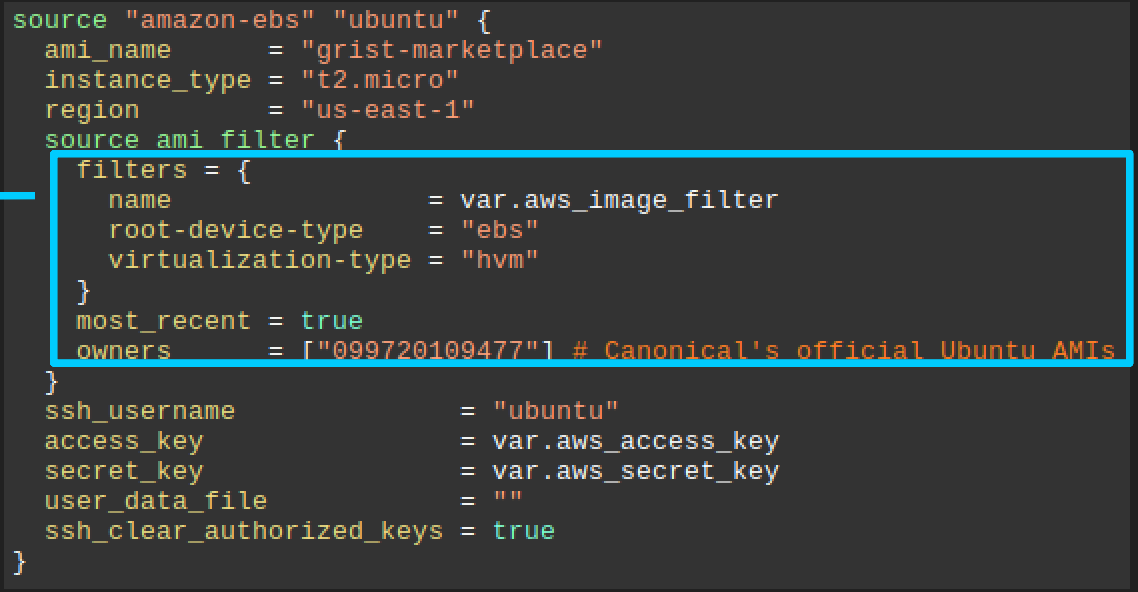
```
source "amazon-ebs" "ubuntu" {
  ami_name      = "grist-marketplace"
  instance_type = "t2.micro"
  region        = "us-east-1"
  source_ami_filter {
    filters = {
      name                = var.aws_image_filter
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"] # Canonical's official Ubuntu AMIs
  }
  ssh_username              = "ubuntu"
  access_key                = var.aws_access_key
  secret_key                = var.aws_secret_key
  user_data_file            = ""
  ssh_clear_authorized_keys = true
}
```

The `grist.pkr.hcl` file

More details on configuring sources

- Use those variables to define the full source block

Where to find base AMI

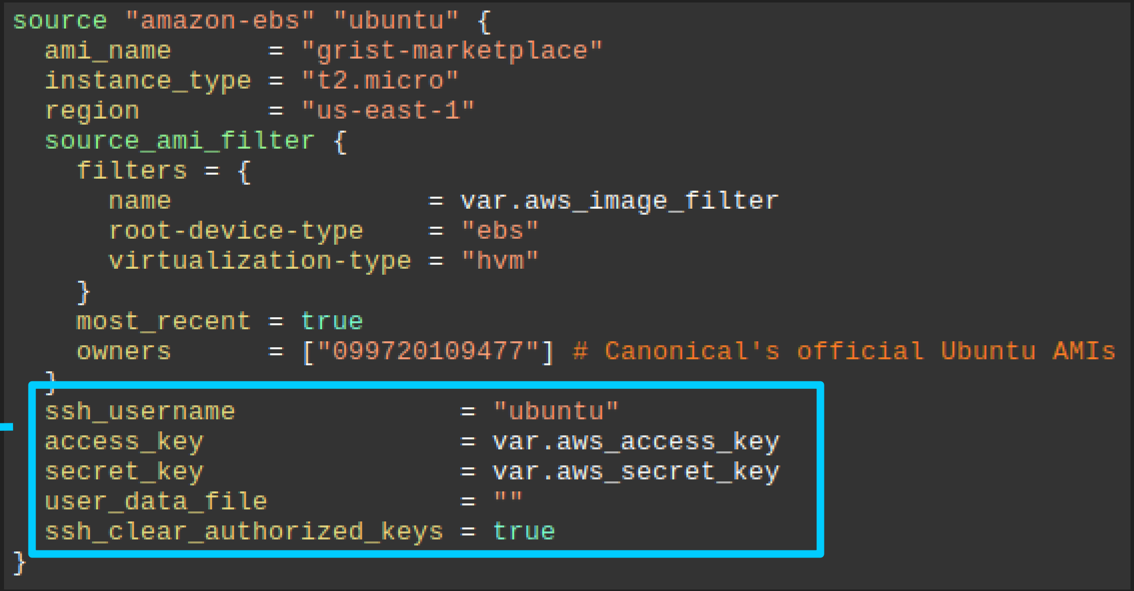


```hcl
source "amazon-ebs" "ubuntu" {
  ami_name      = "grist-marketplace"
  instance_type = "t2.micro"
  region        = "us-east-1"
  source_ami_filter {
    filters = {
      name                = var.aws_image_filter
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"] # Canonical's official Ubuntu AMIs
  }
  ssh_username              = "ubuntu"
  access_key                = var.aws_access_key
  secret_key                = var.aws_secret_key
  user_data_file            = ""
  ssh_clear_authorized_keys = true
}
```

The grist.pkr.hcl file

More details on configuring sources

- Use those variables to define the full source block



```
source "amazon-ebs" "ubuntu" {
  ami_name      = "grist-marketplace"
  instance_type = "t2.micro"
  region        = "us-east-1"
  source_ami_filter {
    filters = {
      name                = var.aws_image_filter
      root-device-type    = "ebs"
      virtualization-type = "hvm"
    }
    most_recent = true
    owners      = ["099720109477"] # Canonical's official Ubuntu AMIs
  }
  ssh_username              = "ubuntu"
  access_key                = var.aws_access_key
  secret_key                = var.aws_secret_key
  user_data_file            = ""
  ssh_clear_authorized_keys = true
}
```

How to auth into AWS

One final ingredient

Assigning values to static variables

One final ingredient

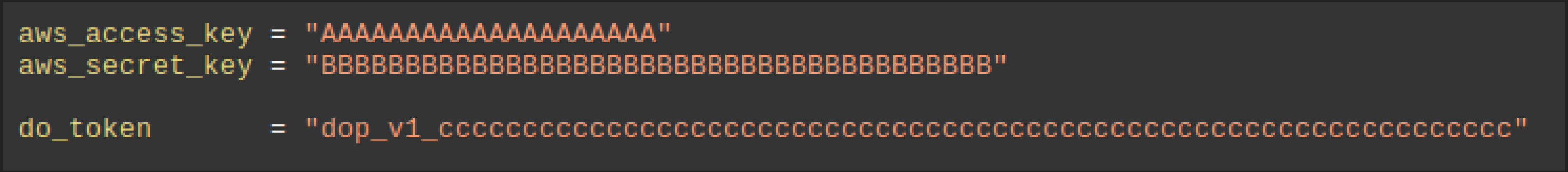
Assigning values to static variables

- Possible to give them values at command line

One final ingredient

Assigning values to static variables

- Possible to give them values at command line
- Or stick them into an untracked `grist.auto.pkrvars.hcl` file



```
aws_access_key = "AAAAAAAAAAAAAAAAAAAA"
aws_secret_key = "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB"

do_token       = "dop_v1_cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc"
```

Quick jargon recap

Basic ingredients of a Packer file

Quick jargon recap

Basic ingredients of a Packer file

- *Plugins* to define build types (amazon, digitalocean, ...)

Quick jargon recap

Basic ingredients of a Packer file

- *Plugins* to define build types (amazon, digitalocean, ...)
- *Builder sources* using those plugins

Quick jargon recap

Basic ingredients of a Packer file

- *Plugins* to define build types (amazon, digitalocean, ...)
- *Builder sources* using those plugins
  - Use *static variables* for builder sources

Quick jargon recap

Basic ingredients of a Packer file

- *Plugins* to define build types (amazon, digitalocean, ...)
- *Builder sources* using those plugins
  - Use *static variables* for builder sources
- A *build block* using these sources

Quick jargon recap

Basic ingredients of a Packer file

- *Plugins* to define build types (amazon, digitalocean, ...)
- *Builder sources* using those plugins
    - Use *static variables* for builder sources
- A *build block* using these sources
    - The build block calls *provisioners* to do the actual work

Running Packer

First init Packer

Running Packer

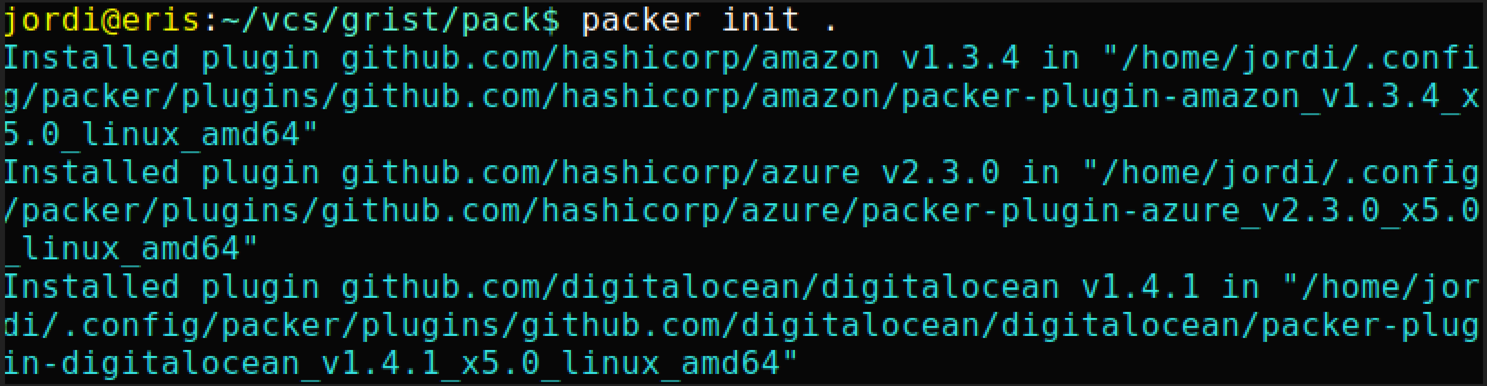
First init Packer

- Run `packer fmt .` (prettifies `.hcl` file)

Running Packer

First init Packer

- Run `packer fmt .` (prettifies `.hcl` file)
- Run `packer init .` (installs plugins)



```
jordi@eris:~/vcs/grist/pack$ packer init .
Installed plugin github.com/hashicorp/amazon v1.3.4 in "/home/jordi/.confi
g/packer/plugins/github.com/hashicorp/amazon/packer-plugin-amazon_v1.3.4_x
5.0_linux_amd64"
Installed plugin github.com/hashicorp/azure v2.3.0 in "/home/jordi/.config
/packer/plugins/github.com/hashicorp/azure/packer-plugin-azure_v2.3.0_x5.0
_linux_amd64"
Installed plugin github.com/digitalocean/digitalocean v1.4.1 in "/home/jor
di/.config/packer/plugins/github.com/digitalocean/digitalocean/packer-plug
in-digitalocean_v1.4.1_x5.0_linux_amd64"
```

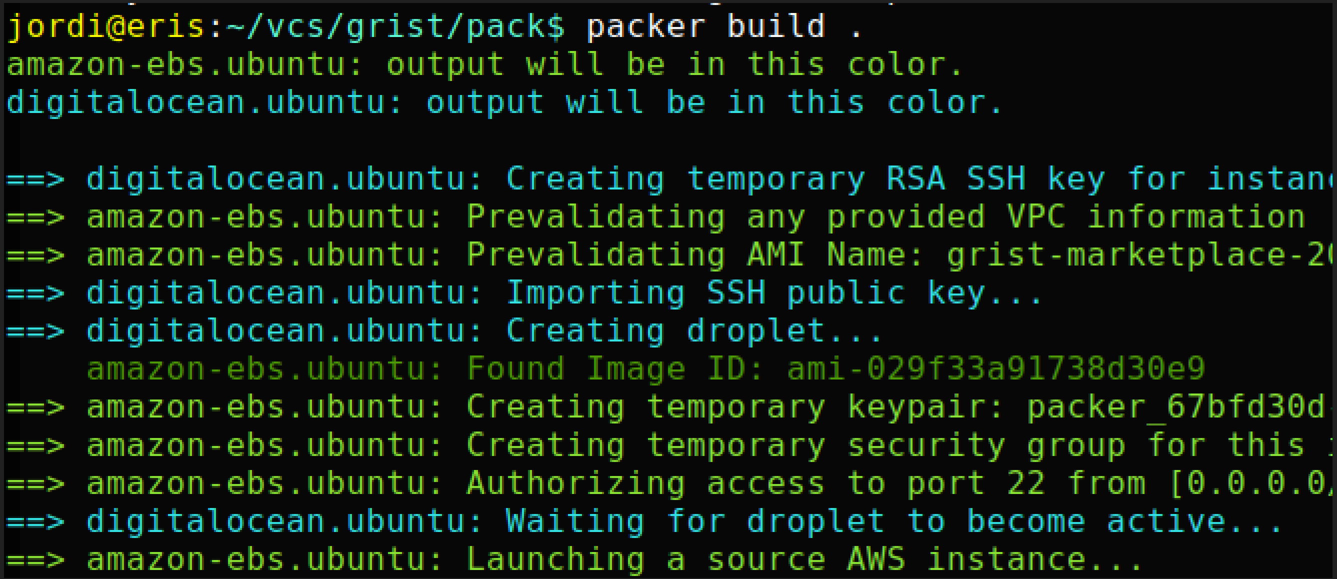
Running Packer

Now give it a spin

Running Packer

Now give it a spin

- Run `packer build .`



```
jordi@eris:~/vcs/grist/pack$ packer build .
amazon-ebs.ubuntu: output will be in this color.
digitalocean.ubuntu: output will be in this color.

==> digitalocean.ubuntu: Creating temporary RSA SSH key for instan
==> amazon-ebs.ubuntu: Prevalidating any provided VPC information
==> amazon-ebs.ubuntu: Prevalidating AMI Name: grist-marketplace-2
==> digitalocean.ubuntu: Importing SSH public key...
==> digitalocean.ubuntu: Creating droplet...
    amazon-ebs.ubuntu: Found Image ID: ami-029f33a91738d30e9
==> amazon-ebs.ubuntu: Creating temporary keypair: packer_67bfd30d
==> amazon-ebs.ubuntu: Creating temporary security group for this
==> amazon-ebs.ubuntu: Authorizing access to port 22 from [0.0.0.0
==> digitalocean.ubuntu: Waiting for droplet to become active...
==> amazon-ebs.ubuntu: Launching a source AWS instance...
```

Those were the basics…

Those were the basics…

… now let’s get more advanced

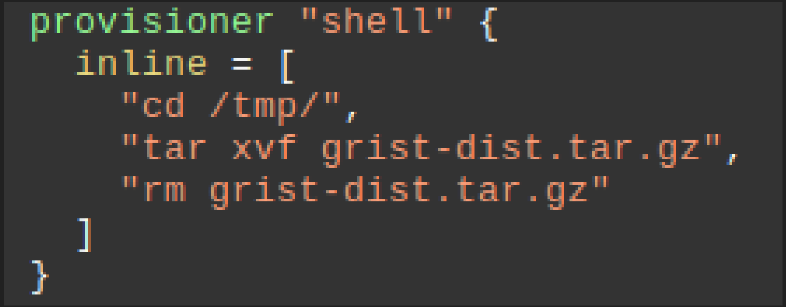
Other configuration details

Kinds of provisioners

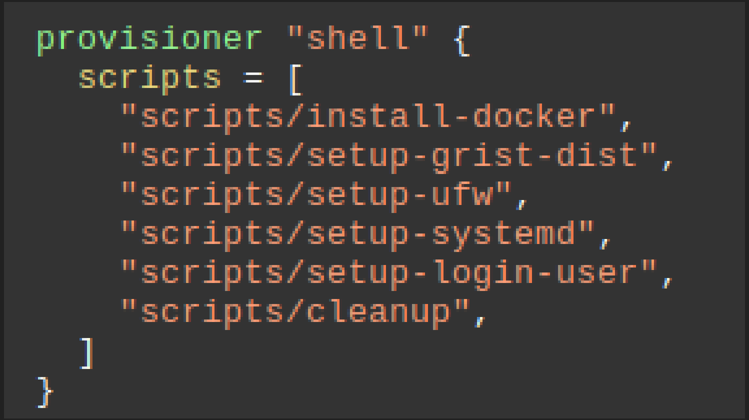
Other configuration details

Kinds of provisioners

- You may use shell scripts



```
provisioner "shell" {
  inline = [
    "cd /tmp/",
    "tar xvf grist-dist.tar.gz",
    "rm grist-dist.tar.gz"
  ]
}
```



```
provisioner "shell" {
  scripts = [
    "scripts/install-docker",
    "scripts/setup-grist-dist",
    "scripts/setup-ufw",
    "scripts/setup-systemd",
    "scripts/setup-login-user",
    "scripts/cleanup",
  ]
}
```

Other configuration details

Kinds of provisioners

- You may use shell scripts
- But there are other options:
  - Ansible
  - Salt
  - Puppet (unmaintained plugin)

Other configuration details

Other possible post-processors

Other configuration details

Other possible post-processors

- Besides building manifests we can

Other configuration details

Other possible post-processors

- Besides building manifests we can
  - Run a local script (shell-local)

Other configuration details

Other possible post-processors

- Besides building manifests we can
  - Run a local script (shell-local)
  - Convert image to local Vagrant (vagrant)

Other configuration details

Other possible post-processors

- Besides building manifests we can
  - Run a local script (shell-local)
  - Convert image to local Vagrant (vagrant)
  - Send the image to CI/CD

Other configuration details

Enabling or disabling sources

Other configuration details

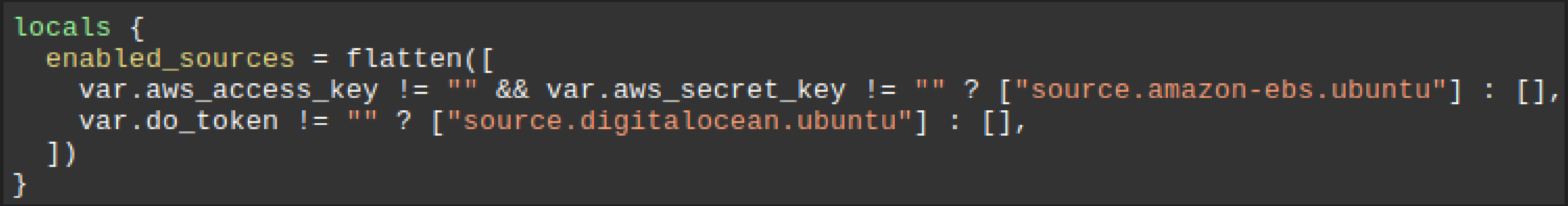
Enabling or disabling sources

- Maybe you don’t always want to build for all cloud providers

Other configuration details

Enabling or disabling sources

- Maybe you don’t always want to build for all cloud providers
- Use dynamic *local variables* to check for credentials

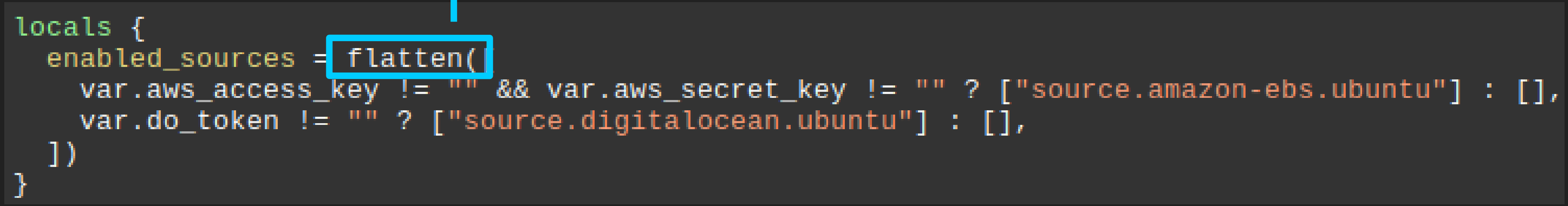


```
locals {
  enabled_sources = flatten([
    var.aws_access_key != "" && var.aws_secret_key != "" ? ["source.amazon-ebs.ubuntu"] : [],
    var.do_token != "" ? ["source.digitalocean.ubuntu"] : [],
  ])
}
```

Other configuration details

Enabling or disabling sources

- Maybe you don’t always want to build for all cloud providers
- Use dynamic *local variables* to check for credentials
  - Note that locals allow HCL function calls

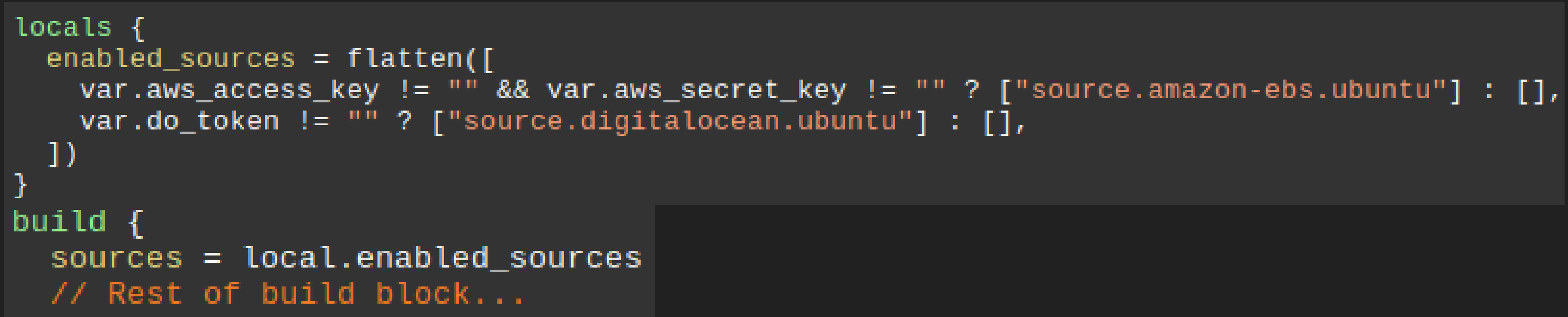


```
locals {
  enabled_sources = flatten([
    var.aws_access_key != "" && var.aws_secret_key != "" ? ["source.amazon-ebs.ubuntu"] : [],
    var.do_token != "" ? ["source.digitalocean.ubuntu"] : [],
  ])
}
```

Other configuration details

Enabling or disabling sources

- Maybe you don’t always want to build for all cloud providers
- Use dynamic *local variables* to check for credentials
  - Note that locals allow HCL function calls
- Use the dynamic list as your build sources



```
locals {
  enabled_sources = flatten([
    var.aws_access_key != "" && var.aws_secret_key != "" ? ["source.amazon-ebs.ubuntu"] : [],
    var.do_token != "" ? ["source.digitalocean.ubuntu"] : [],
  ])
}
build {
  sources = local.enabled_sources
  // Rest of build block...
```

Some useful tricks

Building a tarball payload

Some useful tricks

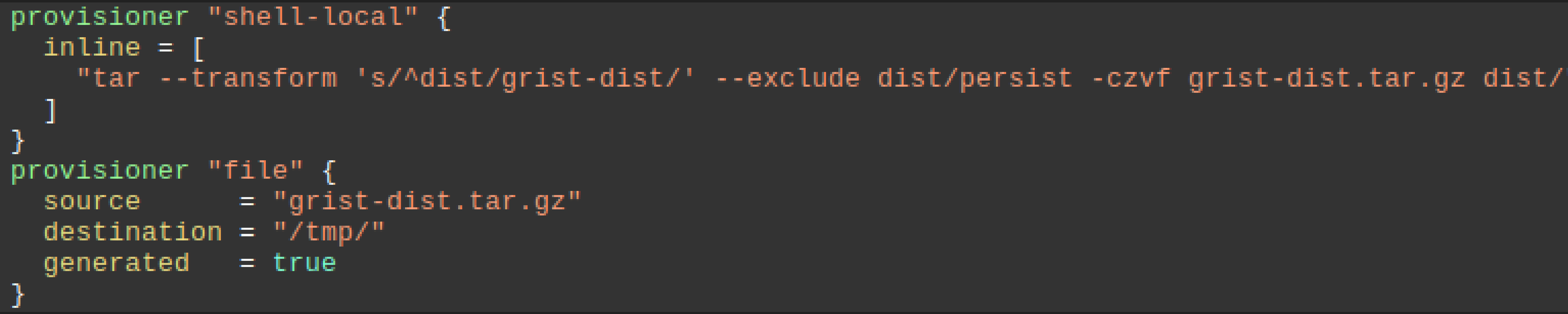
Building a tarball payload

- What if you need to build a file during provisioning?

Some useful tricks

Building a tarball payload

- What if you need to build a file during provisioning?
- Example: build a tarball that excludes some files

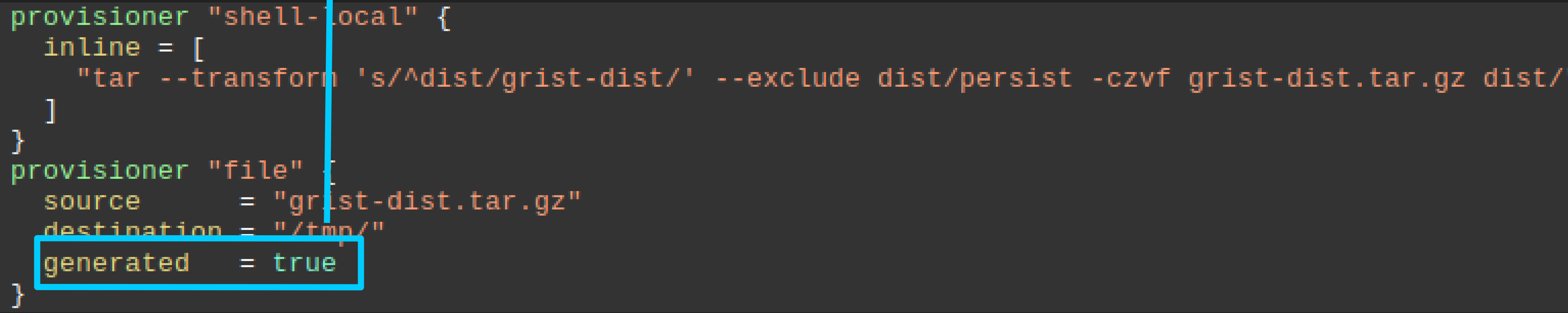


```
provisioner "shell-local" {
  inline = [
    "tar --transform 's/^dist/grist-dist/' --exclude dist/persist -czvf grist-dist.tar.gz dist/
  ]
}
provisioner "file" {
  source      = "grist-dist.tar.gz"
  destination = "/tmp/"
  generated   = true
}
```

Some useful tricks

Building a tarball payload

- What if you need to build a file during provisioning?
- Example: build a tarball that excludes some files
- Use the `generated=true` property to allow dynamic generation



```
provisioner "shell-local" {
  inline = [
    "tar --transform 's/^dist/grist-dist/' --exclude dist/persist -czvf grist-dist.tar.gz dist/
  ]
}
provisioner "file" {
  source      = "grist-dist.tar.gz"
  destination = "/tmp/"
  generated   = true
}
```

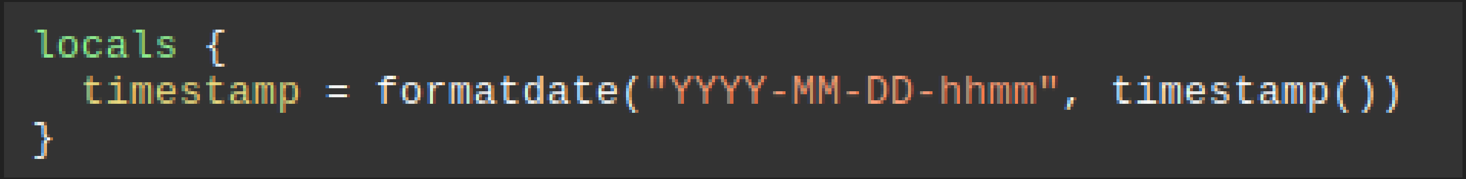
Some useful tricks

Adding a timestamp to machine image name

Some useful tricks

Adding a timestamp to machine image name

- Use local variables again



```
locals {
  timestamp = formatdate("YYYY-MM-DD-hhmm", timestamp())
}
```

Some useful tricks

Adding a timestamp to machine image name

- Use local variables again
- Then interpolate that variable into the image name

```
locals {
  timestamp = formatdate("YYYY-MM-DD-hhmm", timestamp())
}
```

```
source "amazon-ebs" "ubuntu" {
  ami_name       = "grist-marketplace-${local.timestamp}"
  // Rest of source block...
}
```

Some useful tricks

Limiting providers per source

Some useful tricks

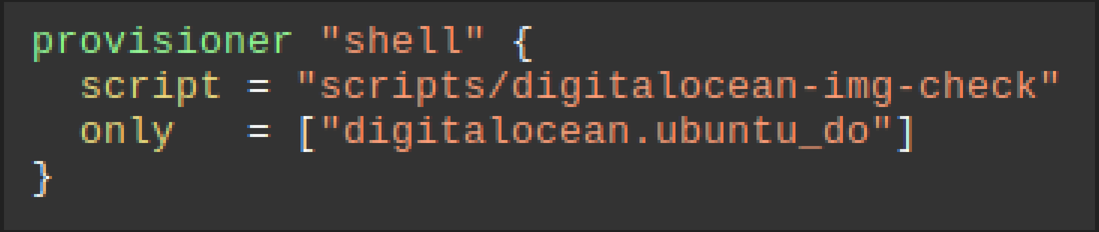
Limiting providers per source

- What if Digital Ocean has a particular provisioning need?

Some useful tricks

Limiting providers per source

- What if Digital Ocean has a particular provisioning need?
- Give it a provisioner

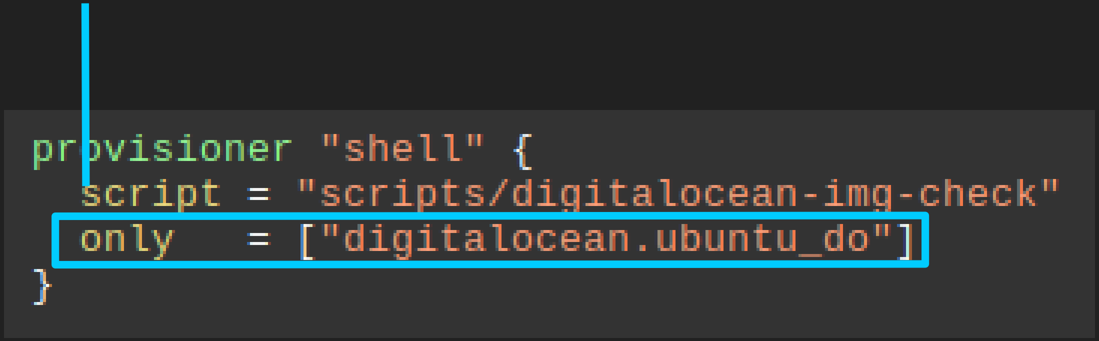


```
provisioner "shell" {
  script = "scripts/digitalocean-img-check"
  only   = ["digitalocean.ubuntu_do"]
}
```

Some useful tricks

Limiting providers per source

- What if Digital Ocean has a particular provisioning need?
- Give it a provisioner
- Use `only` clause so this provisioner only applies to Digital Ocean



```
provisioner "shell" {
  script = "scripts/digitalocean-img-check"
  only   = ["digitalocean.ubuntu_do"]
}
```

Some useful tricks

Handling different default permissions per source

Some useful tricks

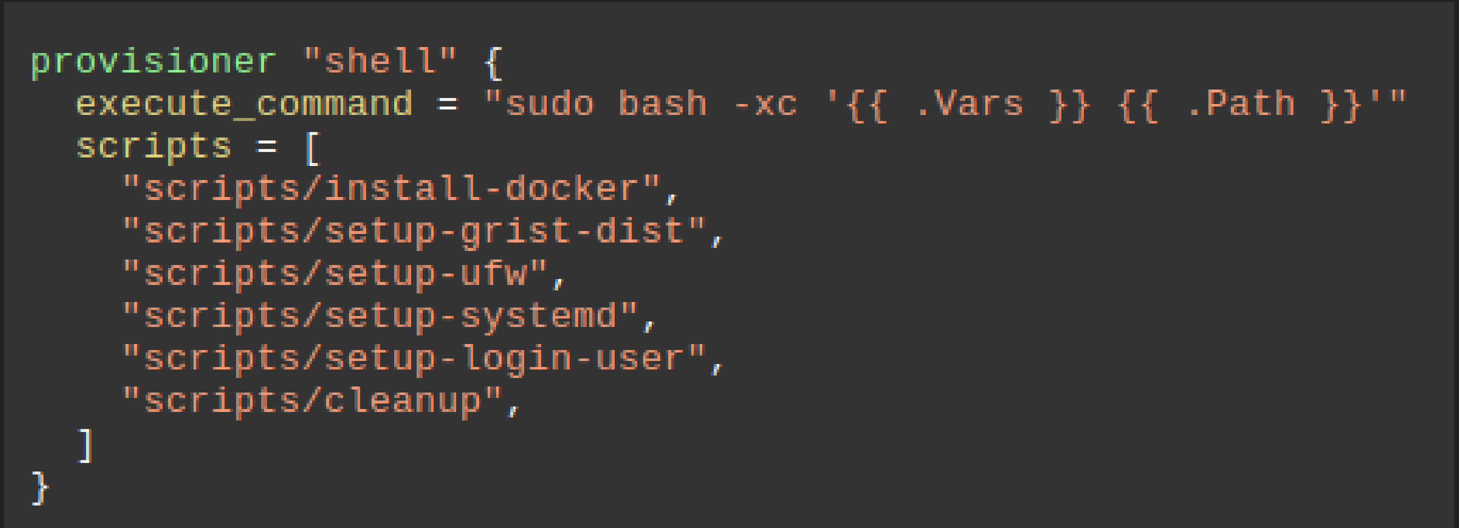
Handling different default permissions per source

- Digital Ocean’s base AMI allows root ssh; AWS does not

Some useful tricks

Handling different default permissions per source

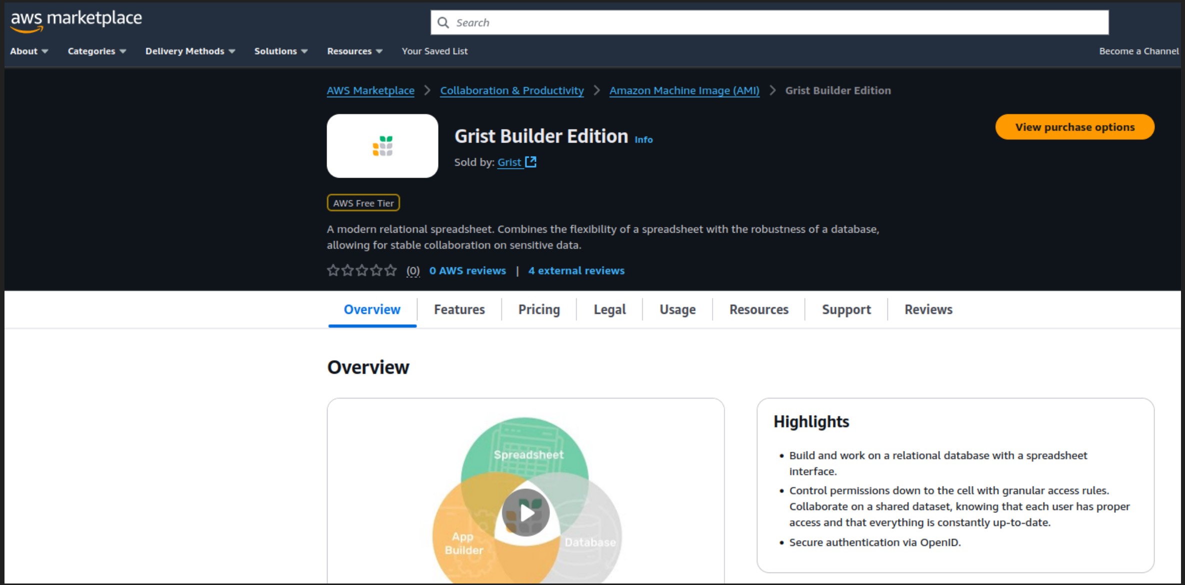
- Digital Ocean’s base AMI allows root ssh; AWS does not
- Tell the provisioner to use `sudo` for both



```
provisioner "shell" {
  execute_command = "sudo bash -xc '{{ .Vars }} {{ .Path }}'"
  scripts = [
    "scripts/install-docker",
    "scripts/setup-grist-dist",
    "scripts/setup-ufw",
    "scripts/setup-systemd",
    "scripts/setup-login-user",
    "scripts/cleanup",
  ]
}
```

After all that hard work…

And we have Grist on the AWS marketplace



AWS Marketplace > Collaboration & Productivity > Amazon Machine Image (AMI) > Grist Builder Edition

Grist Builder Edition

Sold by: Grist

AWS Free Tier

A modern relational spreadsheet. Combines the flexibility of a spreadsheet with the robustness of a database, allowing for stable collaboration on sensitive data.

☆☆☆☆☆ (0) 0 AWS reviews | 4 external reviews

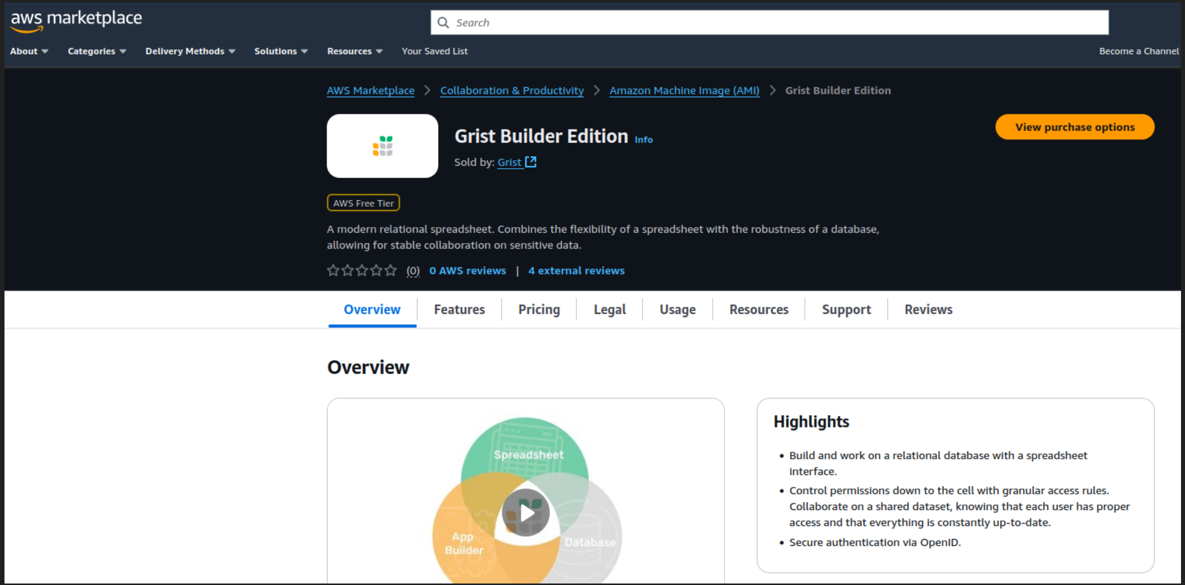
Overview

Highlights

- Build and work on a relational database with a spreadsheet interface.
- Control permissions down to the cell with granular access rules. Collaborate on a shared dataset, knowing that each user has proper access and that everything is constantly up-to-date.
- Secure authentication via OpenID.

And we have Grist on the AWS marketplace

check it out!



Grist Builder Edition

Sold by: Grist

A modern relational spreadsheet. Combines the flexibility of a spreadsheet with the robustness of a database, allowing for stable collaboration on sensitive data.

Overview

Highlights

- Build and work on a relational database with a spreadsheet interface.
- Control permissions down to the cell with granular access rules. Collaborate on a shared dataset, knowing that each user has proper access and that everything is constantly up-to-date.
- Secure authentication via OpenID.

Thank you!

Email: jordi@getgrist.com

Fediverse: @jordigh@mathstodon.xzy