

# Human-Friendly Configuration with Perl 6

or How I Learned to Stop Worrying and Love  
the JSON

Michael Heironimus

# A few words about me

- Infrastructure
- The home for your applications!
- Or sometimes the sausage maker.



# ... it's for

- Tools and automation
- Annoyance reduction
- Team enlightenment

# Packer

Open source from HashiCorp, just like  
Vagrant

Create disk images, Vagrant boxes,  
Docker images

<http://packer.io/>

# How does packer work?

Install VM from ISO or import a disk image

Run setup scripts and tools

Run optional post-install tasks

Configured by handwritten JSON file

# JSON is machine-friendly, not human-friendly

Strict syntax even though it's mostly text

No comments, just app-specific keys for  
annotations

CentOS 6/7 build file is 169 lines and  
5.5k

```
{
  "description": "Build Vagrant-ready CentOS images that look like the of
  "builders": [
    {
      "name": "centos-6-virtualbox",
      "type": "virtualbox-iso",
      "boot_command": [
        "<tab> text {{user `c6_ks_opts`}} ks=http://{{.HTTPIP}}:{{.HTTTPo
      ],
      "disk_size": 8192,
      "export_opts": ["--manifest"],
      "guest_additions_path": "VBoxGuestAdditions_{{.Version}}.iso",
      "guest_os_type": "RedHat_64",
      "hard_drive_interface": "sata",
      "headless": "{{user `headless`}}",
      "http_directory": "http",
      "iso_checksum": "{{user `c6_iso_checksum`}}",
      "iso_checksum_type": "sha512",
      "iso_url": "{{user `c6_iso_path`}}",
      "output_directory": "builds/{{build_name}}-{{user `stamp`}}",
      "shutdown_command": "echo 'vagrant' | sudo -S /sbin/halt -h -p",
      "ssh_password": "vagrant",
      "ssh_username": "vagrant",
      "ssh_wait_timeout": "10000s",
      "vboxmanage": [
```



```

    ["modifyvm", "{{.Name}}", "--memory", "1280"],
    ["modifyvm", "{{.Name}}", "--cpus", "1"],
    ["modifyvm", "{{.Name}}", "--vram", "12"]
  ],
  "vm_name": "{{user `c6_name`}}-{{user `stamp`}}"
},
{
  "name": "centos-7-virtualbox",
  "type": "virtualbox-iso",
  "boot_command": [
    "<tab> text {{user `c7_ks_opts`}} ks=http://{{.HTTPIP}}:{{.HTTTPo
  ],
  "disk_size": 8192,
  "export_opts": ["--manifest"],
  "guest_additions_path": "VBoxGuestAdditions_{{.Version}}.iso",
  "guest_os_type": "RedHat_64",
  "hard_drive_interface": "sata",
  "headless": "{{user `headless`}}",
  "http_directory": "http",
  "iso_checksum": "{{user `c7_iso_checksum`}}",
  "iso_checksum_type": "sha512",
  "iso_url": "{{user `c7_iso_path`}}",
  "output_directory": "builds/{{build_name}}-{{user `stamp`}}",
  "shutdown_command": "echo 'vagrant' | sudo -S /sbin/halt -h -p",
  "ssh_password": "vagrant",

```

```

    "ssh_username": "vagrant",
    "ssh_wait_timeout": "10000s",
    "vboxmanage": [
        ["modifyvm", "{{.Name}}", "--memory", "1280"],
        ["modifyvm", "{{.Name}}", "--cpus", "1"],
        ["modifyvm", "{{.Name}}", "--vram", "12"]
    ],
    "vm_name": "{{user `c7_name`}}-{{user `stamp`}}",
},
{
    "type": "docker",
    "name": "centos-6",
    "commit": true,
    "image": "{{user `c6_docker_image`}}",
    "pull": false
},
{
    "type": "docker",
    "name": "centos-7",
    "commit": true,
    "image": "{{user `c7_docker_image`}}",
    "pull": false
}
],
"post-processors": [

```

```
{
  "type": "vagrant",
  "only": ["centos-6-virtualbox", "centos-7-virtualbox"],
  "compression_level": 9,
  "keep_input_artifact": true,
  "output": "builds/{{.BuildName}}-{{user `stamp`}}.box"
},
[
  {
    "type": "docker-tag",
    "only": ["centos-6", "centos-7"],
    "repository": "mkheironimus/{{build_name}}",
    "tag": "{{user `stamp`}}"
  },
  {
    "type": "docker-save",
    "only": ["centos-6", "centos-7"],
    "path": "builds/{{build_name}}-{{user `stamp`}}.tar"
  }
]
],
"provisioners": [
  {
    "type": "shell",
    "environment_vars": [
```

```

    "HOME_DIR=/home/vagrant",
    "http_proxy={{user `http_proxy`}}",
    "https_proxy={{user `https_proxy`}}",
    "no_proxy={{user `no_proxy`}}",
    "VERBOSE={{user `verbose`}}"
  ],
  "override": {
    "centos-6-virtualbox": {
      "execute_command": "env {{.Vars}} sudo -S -E bash -eu '{{.Path}}'",
    },
    "centos-7-virtualbox": {
      "execute_command": "env {{.Vars}} sudo -S -E bash -eu '{{.Path}}'"
    }
  },
  "execute_command": "env {{.Vars}} bash -eu '{{.Path}}'",
  "scripts": [
    "shell/yum.sh",
    "shell/docker_prep.sh",
    "shell/grub.sh",
    "shell/patch.sh"
  ]
},
{
  "type": "shell",
  "pause_before": "10s",

```

```

    "environment_vars": [
        "HOME_DIR=/home/vagrant",
        "http_proxy={{user `http_proxy`}}",
        "https_proxy={{user `https_proxy`}}",
        "no_proxy={{user `no_proxy`}}",
        "VERBOSE={{user `verbose`}}"
    ],
    "override": {
        "centos-6-virtualbox": {
            "execute_command": "env {{.Vars}} sudo -S -E bash -eu '{{.Path}}'",
        },
        "centos-7-virtualbox": {
            "execute_command": "env {{.Vars}} sudo -S -E bash -eu '{{.Path}}'"
        }
    },
    "execute_command": "env {{.Vars}} bash -eu '{{.Path}}'",
    "scripts": [
        "shell/vmtools.sh",
        "shell/vagrant.sh",
        "shell/vm_ami_cleanup.sh",
        "shell/cleanup.sh",
        "shell/whiteout.sh"
    ]
}
],

```

```
"variables": {  
  "headless": "true",  
  "http_proxy": "{{env `http_proxy`}}",  
  "https_proxy": "{{env `https_proxy`}}",  
  "no_proxy": "{{env `no_proxy`}}",  
  "stamp": "{{isotime \"20060102\"}}",  
  "verbose": "true",  
  "c6_name": "centos-6",  
  "c6_iso_checksum": "c694ba4903ac2eb07d996ffa06b92d1dda78ec035d1ad87fd",  
  "c6_iso_path": "iso/CentOS-6.7-x86_64-bin-DVD1.iso",  
  "c6_ks_opts": "",  
  "c6_docker_image": "centos:6",  
  "c7_name": "centos-7",  
  "c7_iso_checksum": "3e084199713f0a7a39942a123ad698d36e0ee0a2253774e96",  
  "c7_iso_path": "iso/CentOS-7-x86_64-DVD-1511.iso",  
  "c7_ks_opts": "rd.live.check=0",  
  "c7_docker_image": "centos:7"  
}  
}
```

# Packer JSON is repetitious

If you put builders in the same file, you  
duplicate builders

If you put builders in different files, you  
duplicate everything else

Supports user-defined variables, but can  
not iterate

# Enter the YAML

Natural starting point: trivially converted  
to/from JSON

Easy syntax, quotes only where needed

Commenting

<http://yaml.org/>



... buuuuut

Nested arrays look ugly

Lots of Packer config still needs quotes

Direct conversion is still repetitious

# DRY it up

(Don't Repeat Yourself)

Include other YAML files

Initial attempt at some top-level global  
settings

```
---
builders:
  centos-6-virtualbox:
    config_template:
      - tpl_vbox.yml
      - tpl_vbox_iso.yml
    iso_url: iso/CentOS-6.7-x86_64-bin-DVD1.iso
    boot_command:
      - <tab> text ks=http://{{.HTTPIP}}:{{.HTTPPort}}/centos-6.ks<enter>
    vm_name: "centos-6-{{user `stamp`}}"
    iso_checksum: c694ba4903ac2eb07d996ffa06b92d1dda78ec035d1ad87fdc9681e
  centos-7-virtualbox:
    config_template:
      - tpl_vbox.yml
      - tpl_vbox_iso.yml
    iso_url: iso/CentOS-7-x86_64-DVD-1511.iso
    boot_command:
      - <tab> text rd.live.check=0 ks=http://{{.HTTPIP}}:{{.HTTPPort}}/ce
    vm_name: "centos-7-{{user `stamp`}}"
    iso_checksum: 3e084199713f0a7a39942a123ad698d36e0ee0a2253774e96c44a85
  centos-6:
    config_template: tpl_docker_commit.yml
    image: centos:6
  centos-7:
    config_template: tpl_docker_commit.yml
```

```
    image: centos:7
  provisioner_override:
    centos-6:
      execute_command: "env {{.Vars}} bash -eu '{{.Path}}'"
    centos-7:
      execute_command: "env {{.Vars}} bash -eu '{{.Path}}'"
  provisioners:
  -
    config_template: tpl_shell.yml
    scripts:
    - shell/yum.sh
    - shell/docker_prep.sh
    - shell/grub.sh
    - shell/patch.sh
  -
    config_template: tpl_shell.yml
    scripts:
    - shell/vmtools.sh
    - shell/vagrant.sh
    - shell/vm_ami_cleanup.sh
    - shell/cleanup.sh
    - shell/whiteout.sh
    pause_before: 10s
  post-processors:
  -
```

```
    config_template: tpl_vagrant.yml
  only:
    - centos-6-virtualbox
    - centos-7-virtualbox
-
-
  config_template: tpl_docker_tag.yml
  only:
    - centos-6
    - centos-7
-
  config_template: tpl_docker_save.yml
  only:
    - centos-6
    - centos-7
# defaults, to be overridden on command line or var file.
variables:
  headless: "true"
  http_proxy: "{{env `http_proxy`}}"
  https_proxy: "{{env `https_proxy`}}"
  no_proxy: "{{env `no_proxy`}}"
  stamp: "{{isotime \"20060102\"}}"
  verbose: "true"
```

# Builder Example

```
centos-6-virtualbox:
  config_template:
    - tpl_vbox.yml
    - tpl_vbox_iso.yml
  iso_url: iso/CentOS-6.7-x86_64-b
  boot_command:
    - <tab> text ks=http://{{.HTTP
  vm_name: "centos-6-{{user `stamp
  iso_checksum: c694ba4903ac2eb07d
```

```
output_directory: "builds/{{build_name}}-{{user `stamp`  
ssh_password: vagrant  
ssh_wait_timeout: 10000s  
headless: "{{user `headless`}}"  
export_opts:  
  - --manifest  
ssh_username: vagrant  
vboxmanage:  
  -  
    - modifyvm  
    - "{{.Name}}"  
    - --memory  
    - "1280"  
  -  
    - modifyvm  
    - "{{.Name}}"  
    - --cpus  
    - "1"  
  -  
    - modifyvm  
    - "{{.Name}}"  
    - --vram  
    - -- ..
```

```
# Overlay for tpl_vbox.yml
type: virtualbox-iso
http_directory: http
hard_drive_interface: sata
guest_os_type: RedHat_64
guest_additions_path: VBoxGuestAdd.
disk_size: 8192
iso_checksum_type: sha512
```



```
centos-6-virtualbox:
  config_template:
    - tpl_vbox.yml
    - tpl_vbox_iso.yml
  iso_url: iso/CentOS-6.7-x86_64-b
  boot_command:
    - <tab> text ks=http://{{.HTTP
  vm_name: "centos-6-{{user `stamp
  iso_checksum: c694ba4903ac2eb07d9
```

# The Script

Minimal testing, but It Works For Me  
(TM)

No error handling

- Perl will tell you if files can't be read or written
- YAMLish will complain if the input is not valid  
YAML
- Packer will tell you if the JSON output is wrong

# Perls of Wisdom

Rakudo Star ships with JSON::Tiny

JSON::Pretty gives you formatted output

YAML is unfinished, so use YAMLish

# Modules, you say?

Modules you expect to use may be unfinished

Documentation / examples may be lacking

YAMLish output fails to quote some strings

# Building a YAML starting point

```
use JSON::Pretty;  
use YAMLish;
```

```
my $f = slurp('centos.json');  
my $cfg = from-json($f);  
my $yaml = save-yaml($cfg);  
say $yaml;
```

```
use v6;  
  
use JSON::Pretty;  
use YAMLish;  
  
class PackerGen {  
    has %!cfg = ();  
    has %.output = ();  
  
    # Constructor taking an optional input file, not in  
    submethod BUILD(Str :$yaml='') {  
        self.load($yaml);  
        self.generate();  
    }  
  
    # Load a YAML file from stdin or the named file.  
    method load(Str $in) {  
        %!cfg = load-yaml(($in ne '') ?? slurp($in)  
            !! $*IN.slurp-rest);  
    }  
}
```

```

# Write the generated JSON to stdout or the given file
multi method save(Str $out where { $out eq '' }) {
    say to-json(%!output);
}

multi method save(Str $out) {
    # to-json doesn't have a final newline
    spurt($out, to-json(%!output) ~ "\n");
}

# If a config_template element exists, load that config
# overlay the supplied data. Otherwise just return the data
multi method expand_entry(%data
    where { %data<config_template>:exists } -->
    my %new = %data<config_template>:delete.map({
        load-yaml(slurp($^a)) });
    %(%new, %data);
}

multi method expand_entry(%data --> Hash) {
    %data;
}

```

```

# Process templates and add the name key.
method build(Pair (:key(:$name), :value(:$data)) --
    %(self.expand_entry($data), 'name' => $name);
}

```

```

# Process templates and add the global override def
# there is no override in the provisioner.

```

```

method provision(%prov --> Hash) {
    my %new = self.expand_entry(%prov);
    if (!cfg<provisioner_override>:exists &&
        !(%prov<override>:exists)) {
        %new<override> = %!cfg<provisioner_override>
    }
    %new;
}

```

```

# Process templates for a sequence or a single post
multi method postproc(@post --> Seq) {
    @post.map({ self.expand_entry(%^a) });
}

```

--- . . . --- . . . ---



```
method generate(--> Hash) {
  %!output<builders> = %!cfg<builders>.map({
    self.build($^a) });

  %!output<provisioners> = %!cfg<provisioners>.map({
    self.provision($^a) })
    if (%!cfg<provisioners>:exists);

  %!output<post-processors> = %!cfg<post-processors>.map({
    self.postproc($^a) })
    if (%!cfg<post-processors>:exists);

  %!output<description> = %!cfg<description>
    if (%!cfg<description>:exists);

  %!output<variables> = %!cfg<variables>
    if (%!cfg<variables>:exists);

  %!output;
}
```

```
# --in=input.yaml --out=output.json
sub MAIN(Str :$in='', Str :$out='') {
    my $pg = PackerGen.new(yaml => $in);
    $pg.save($out);
}
```

# Many possible futures

Search path for template files

More advanced dynamic templating

Support other tools - AWS and HashiCorp  
love JSON, I don't

Full tool wrapper / DSL where it's useful

# That's all, folks!

<https://github.com/mkheironimus/packergen>

<https://mkheironimus.github.io/slides/20160316-packergen.pdf>