



# Practical z/VM Automation using Ansible and Terraform

Vic Cross, IBM

[viccross@au.ibm.com](mailto:viccross@au.ibm.com)



## Agenda

- What are these tools?
  - Ansible
  - Terraform
  - Packer
- What can be automated?
- Works in progress
  - z/VM ESI
  - Packer/Terraform demo



# Introduction to the tools

- Ansible
  - Cross/multi platform configuration management
  - “Idempotence”
  - Ideal for post-deployment customization
  - Commercial product: Ansible Automation Platform





- YAML
- Jinja2 (for templating)
- Can use JSON data objects

# What does it look like?

```
# Snippet from Ansible config: syntax example only
```

```
- name: Create DNS Forward zone (internal)
  template:
    src: var/named/ibmpoc_internal.zone.j2
    dest: /var/named/ibmpoc_internal.zone
    owner: named
    group: named
    mode: 0640
  notify:
    - restart dns

- name: unmanage resolve.conf in NetworkManager
  lineinfile:
    line: dns=none
    dest: /etc/NetworkManager/NetworkManager.conf
    insertafter: "\\[main\\].*"
    regexp: "^dns=.*"
```



# Introduction to the tools

## Terraform

- “Infrastructure-as-Code”
- Manages state of resources
- “Providers” for wide variety of cloud and infra installations
- Commercial: Terraform Enterprise

## Terraform

**Enterprise-ready from day one**  
Adopt turnkey solutions with enterprise support so you can focus on improving your business outcomes, not IT operations.

**Simplify complex IaC workflows**  
Reuse infrastructure components and automatically enforce security to limit risk and maintain efficiency.

**Lifecycle management at scale**  
Put a system in place to programmatically scale infrastructure, enable self-service, and monitor health over time.

**Code**  
Infrastructure, policies

**Output**  
Infrastructure state, audit logs

HCL

Providers

Cloud | Private datacenter | Service

Infrastructure

14 ©2025 HASHICORP

VM Workshop

5



- HCL
  - HashiCorp Configuration Language

# What does it look like?

```
# Snippet from Terraform config: syntax example only

data "openstack_compute_keypair_v2" "my_keypair" {
  name      = "valhalla-keypair"
}

resource "openstack_compute_instance_v2" "zvm_instance" {
  name          = var.name
  image_id     = data.openstack_images_image_v2.img.id
  flavor_id    = data.openstack_compute_flavor_v2.size.id
  network {
    name = "${data.openstack_networking_network_v2.network.name}"
    fixed_ip_v4 = var.ip_address
  }
  availability_zone = var.availability_zone
  key_pair      = data.openstack_compute_keypair_v2.my_keypair.name
}
```



# Introduction to the tools

## Packer

- Build images
- Consistency in deployment
- “Sources”
  - similar to Terraform providers

## Packer

### Better together with Terraform

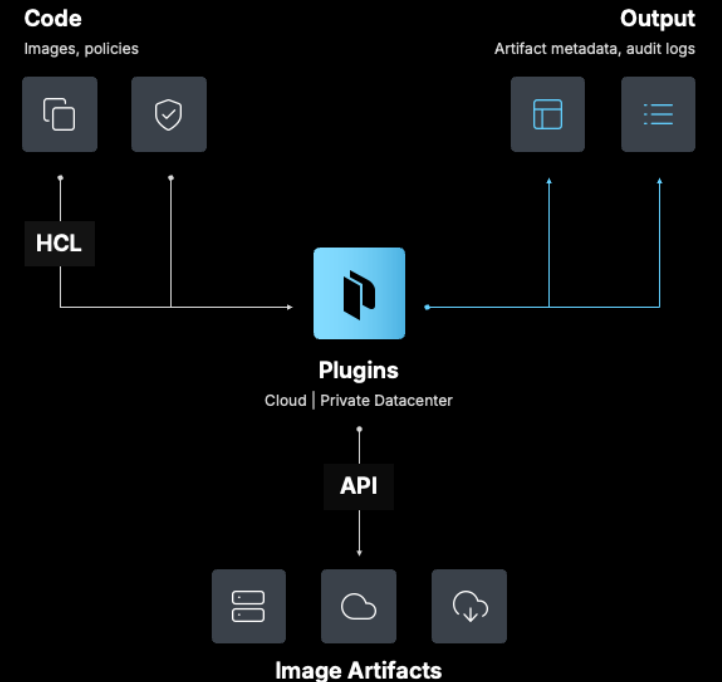
Create a golden image pipeline to unify image and provisioning workflows.

### Lifecycle management

Monitor health and standardize revocation to clean up outdated images posing risk and driving up cloud spend.

### Security and governance

Secure images before deployment to prevent vulnerabilities from occurring in the first place and reduce the window for exploitation.





- HCL
  - HashiCorp Configuration Language

# What does it look like?

# Snippet from Packer config: syntax example only

```
build {
  name = "demo_packer_build"

  provisioner "ansible" {
    playbook_file = "${path.cwd}/ansible/playbook.yml"
    ansible_env_vars = [
      "ANSIBLE_DEPRECATION_WARNINGS=False",
      "ANSIBLE_HOST_KEY_CHECKING=False",
      "ANSIBLE_NOCOLOR=True",
      "ANSIBLE_NOCOWS=1",
      "ANSIBLE_PYTHON_INTERPRETER=/usr/bin/python3",
      "MY_BUILD_NAME=${source.name}"
    ]
    extra_arguments = [
      "--ssh-extra-args",
      "-o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa -o IdentitiesOnly=yes"
    ]
  }
}

source "source.openstack.infra_cloud_center" {
  name          = "httpd_build"
  image_name    = local.httpd_build_name
}
}
```



# What can be automated?



- No official z/VM support
- Watch Jay Brenneman's work
  - zvm\_ansible\_collection  
[https://github.com/IBM/zvm\\_ansible\\_collection](https://github.com/IBM/zvm_ansible_collection)
  - Uses zthin from Feilong
  - Relies on SMAPI

## Ansible

- Ansible supports external commands
  - Modules “shell” and “command”
  - Call out to smcli or other tool
  - No idempotence!
    - Watch out for race/dependency conditions
- A lot is possible even with this...



# Terraform

- Uses ICIC to support its OpenStack provider
  - Terraform doesn't know z/VM
- OpenStack provider does a lot!
  - Guest (VM) creation/deletion
  - Network/subnet management
  - More?
- Packer is similar
  - Supports z/VM using OpenStack via ICIC
- Native provider for z/VM?
  - SMAPI (maybe via Feilong)
  - Some OpenStack concepts don't map directly to z/VM concepts
    - “Availability Group”



# What about deploying z/VM itself?

- Both Ansible and Terraform are close
  - Ansible closer (thanks to existing `ibm_zhmc` collection)
    - In Terraform, use Ansible provider to call Ansible to drive the collection
- z/VM ESI automatic installation enables this
  - Ansible playbook:
    - Create the Partition (DPM only: use existing LPAR if Classic IO DF)
    - Create the ESI CSV file from a template
    - Load the Partition/LPAR (DPM, or Classic IO DF z16 or greater)
    - Wait for install to complete, then re-IPL



# Works in progress



# z/VM ESI ELAN automation

- RHOCP installation
  - Creates guests
  - Generates RHOCP Ignition files
  - Configures DNS/HAProxy/Apache/more
  - Logs on/IPLs guests (via SMAPI)
  - Checks progress
  - Reconfigures cluster for LDAP authentication and replacement TLS certificate
  - Example playbooks at <https://github.com/viccross/ocp-z-ansible-sample>
    - Subset of the full function in ESI



# z/VM ESI ELAN automation

- ICIC deployment
  - Combination of shell scripting and Ansible
    - Script to create guests, Ansible to install ICIC
  - Ansible does post-install LDAP authentication and TLS certificate replacement



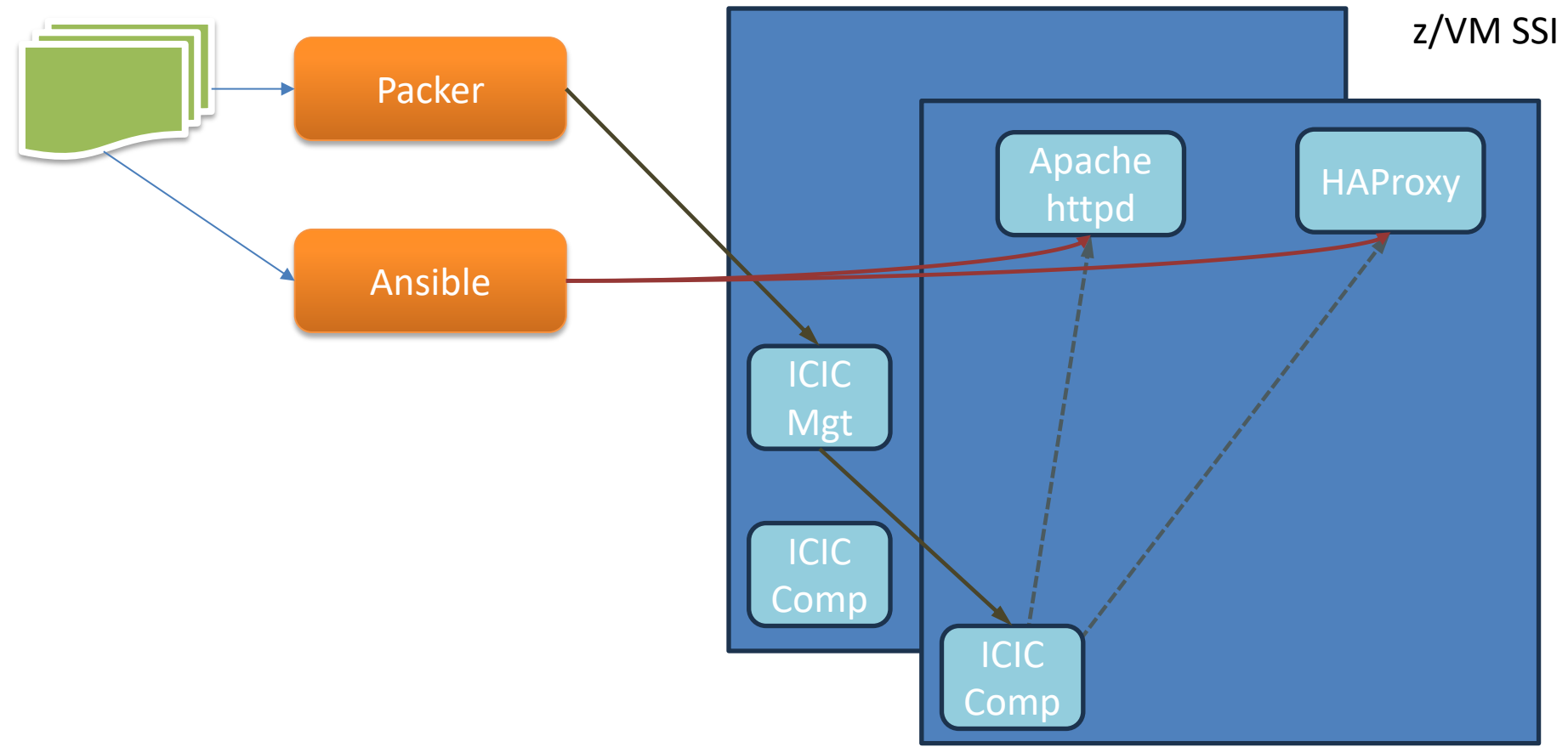
# Terraform with ICIC

- Example use case combining Packer, Terraform, and Ansible
  - Use known base image to create two customized images
    - “Web app” with Apache HTTP Server, PHP-FPM, and a phpinfo page
    - HAProxy (load balancer)
  - Deploy web application serving infrastructure
    - Three HTTP server VMs
    - One load balancer
  - All automatically (and programmatically) deployed



# Terraform with ICIC

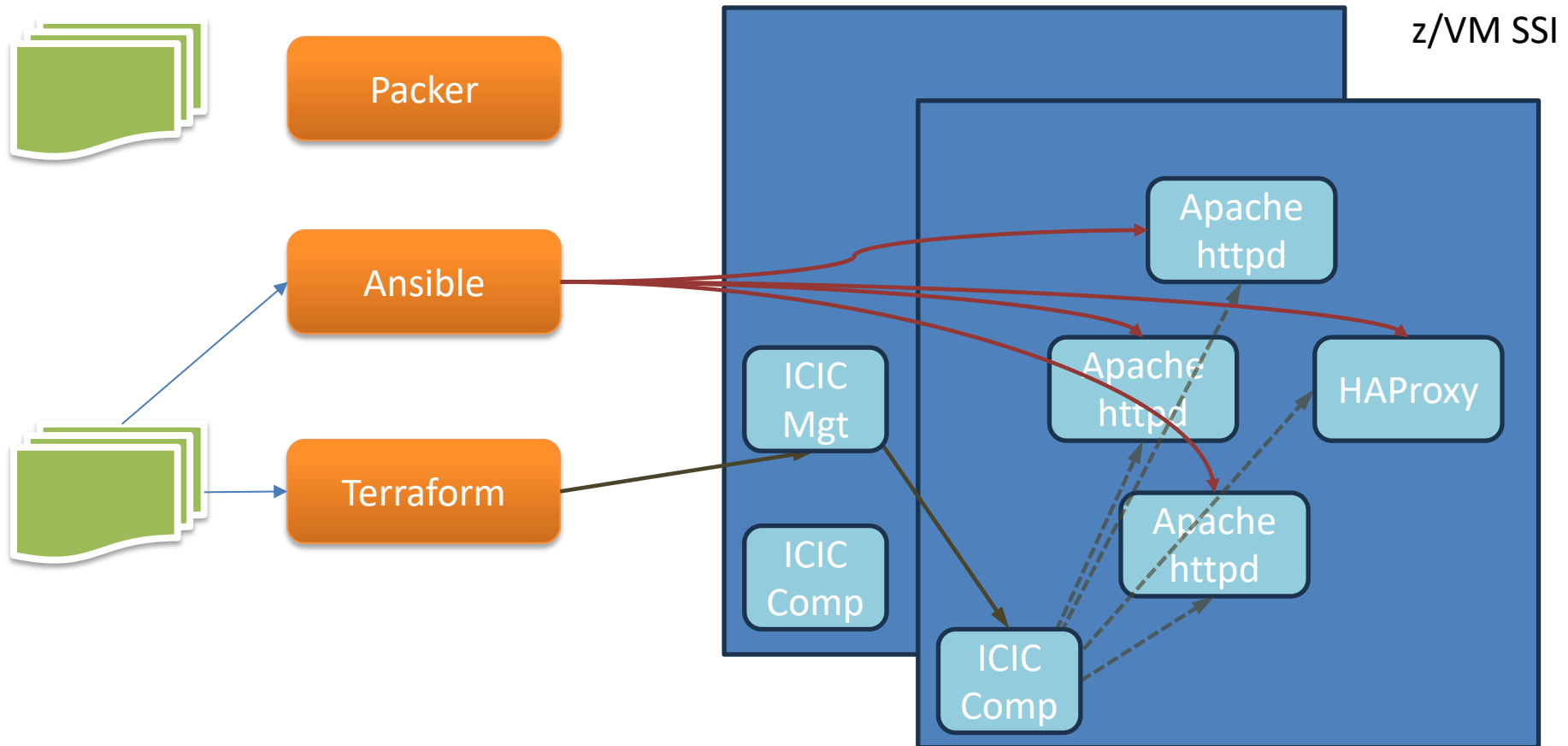
(Phase One: image creation with Packer)





# Terraform with ICIC

(Phase Two: Terraform uses images to make VMs)





# Demo

Demo video coming soon (contact me)!

Try it yourself, if you have ICIC!

Code is at <https://github.com/viccross/icic-packer-terraform-ansible>



# What we don't get time to see

- Terraform responding to changes:
  - In the definition
    - `terraform plan` shows the delta from the known state
  - In the images (e.g. new Packer build)
    - Again, `terraform plan` shows what needs to be rebuilt
      - Automatically based on `most_recent`
      - Manually by changing version control marker in the image



# Where to next?



# Your feedback welcome

- Does this bring back the glory days of Linux cloning, or what!!!
- Integration with CI/CD – treat VMs like containers
  - IaC Terraform/Packer/Ansible in source control
  - Automatic redeploy when the source changes
- OpenStack via ICIC, or native providers for z/VM?



Thank you!

**Vic Cross**

Senior Technical Specialist, zAcceleration

[viccross@au.ibm.com](mailto:viccross@au.ibm.com)