OpenStack

ORCHESTRATION

and

Inside and Out

Florian Haas, florian@hastexo.com



A few



to start with



You should



OpenStack



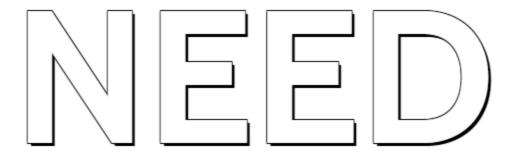
You're welcome to

FOLLOW ALONG

https://github.com/fghaas/osil2015orchestration



You'll



an OpenStack environment



If you're

NOTABLE

or

NOT INCLINED

to follow along,

THAT'S OK.

Just listen and participate in the discussion.



Let's talk about





can we automate

VIRTUAL SYSTEMS

in OpenStack?

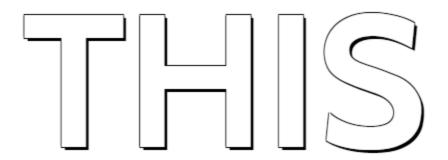
Nova

USER DATA



```
nova boot \
   --image trusty-server-cloudimg-amd64 \
   --key_name mykey \
   --flavor m1.small \
   --user-data userdata.txt \
   --nic net-id=4f0dcc21-4b6c-47db-b283-591fdb9aa5a7 \
   test0
```



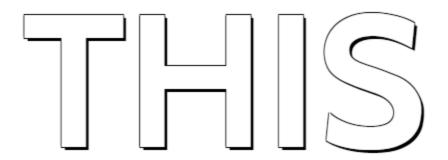


is what user-data



looks like:





is what user-data



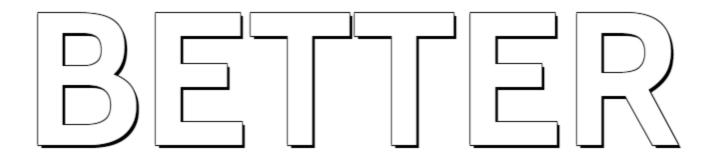
looks like:







You can do





Enter

cloud-config



cloud-config

enables you to

BOOTSTRAP

a newly booted VM



cloud-config is 100%





cloud-config is OpenStack's most

UNDERRATED

feature



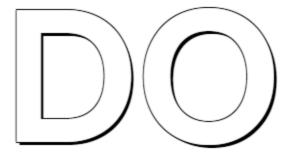
cloud-config is Ubuntu's most

UNDERRATED

feature



What can we



With cloud-config?



package_update package_upgrade

Update system on first boot



#cloud-config

package_update: true
package_upgrade: true



users

Configure users and groups



users: - default - name: foobar gecos: "Fred Otto Oscar Bar" groups: users,adm lock-passwd: false passwd: \$6\$rounds=4096\$J86aZz0Q\$To16RGzWJku0 shell: /bin/bash sudo: "ALL=(ALL) NOPASSWD:ALL"



ssh_pwauth

Enable/disable SSH password authentication



ssh_pwauth: true



write_files

Write arbitrary files



```
write_files:
- path: /etc/hosts
 permissions: '0644'
 content:
   127.0.0.1 localhost
    ::1 ip6-localhost ip6-loopback
   fe00::0 ip6-localnet
   ff00::0 ip6-mcastprefix
   ff02::1 ip6-allnodes
   ff02::2 ip6-allrouters
   192.168.122.100 deploy.example.com deploy
   192.168.122.111 alice.example.com alice
   192.168.122.112 bob.example.com bob
   192.168.122.113 charlie.example.com charlie
```



puppet

Configure a VM's Puppet agent



```
puppet:
 conf:
   agent:
     server: "puppetmaster.example.org"
    certname: "%i.%f"
   ca cert:
     ----BEGIN CERTIFICATE----
    MIICCTCCAXKgAwIBAgIBATANBgkqhkiG9w0BAQUFADANMQswCQYDVQQDDAJjYTAe
     Fw0xMDAyMTUxNzI5MjFaFw0xNTAyMTQxNzI5MjFaMA0xCzAJBgNVBAMMAmNhMIGf
    MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCu7Q40sm47/E1Pf+r8AYb/V/FWGPgc
     b014OmNoX7dgCxTDvps/h8Vw555PdAFsW5+QhsGr31IJNI3kSYprFQcYf7A8tNWu
     SIb3DQEBBQUAA4GBAH/rxlUIjwNb3n7TXJcDJ6MMHUlwjr03BDJXKb34Ulndkpaf
    +GAlzPXWa7bO908M9I8RnPfvtKnteLbvgTK+h+zX1XCty+S2EQWk29i2AdoqOTxb
    hppiGMp0tT5Havu4aceCXiy2crVcudj3NFciy8X66SoECemW9UYDCb9T5D0d
     ----END CERTIFICATE----
```



chef

Configure a VM's Chef client



```
chef:
install_type: "packages"
force_install: false
 server_url: "https://chef.yourorg.com:4000"
 node_name: "your-node-name"
 environment: "production"
validation_name: "yourorg-validator"
validation_key: |
     ----BEGIN RSA PRIVATE KEY-----
    YOUR-ORGS-VALIDATION-KEY-HERE
     ----END RSA PRIVATE KEY----
run_list:
 - "recipe[apache2]"
 - "role[db]"
 initial_attributes:
    apache:
      nrefork.
```



packages

Install packages



packages: - ansible

- git



Running

ARBITRARY COMMANDS



bootemd

Run commands early in the boot sequence



bootcmd:

- ntpdate pool.ntp.org



runcmd

Run commands late in the boot sequence



```
runcmd:
    - >
    sudo -i -u training
    ansible-pull -v -i hosts
    -U https://github.com/hastexo/academy-ansible
    -o site.yml
```



(not an acronym)





enables you to deploy



virtual environments

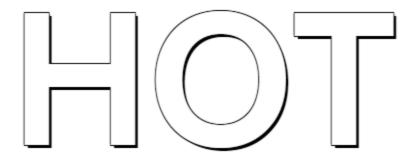


supports two distinct





Amazon CloudFormation compatible template



Heat Orchestration Template

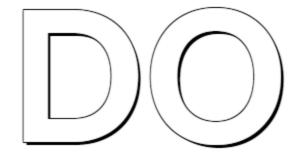


HOT is 100%





What can we



with Heat?



OS::Nova::Server

Configures Nova guests



```
resources:
   mybox:
    type: "OS::Nova::Server"
   properties:
     name: mybox
    image: trusty-server-cloudimg-amd64
    flavor: m1.small
    key_name: mykey
```



Now we could just



this stack



heat stack-create -f stack.yml mystack



But as it is, it's not very





Let's add some

PARAMETERS



```
parameters:
    flavor:
        type: string
        description: Flavor to use for servers
        default: m1.medium
    image:
        type: string
        description: Image name or ID
        default: trusty-server-cloudimg-amd64
    key_name:
        type: string
        description: Keypair to inject into newly created servers
```



And some

INTRINSIC FUNCTIONS



```
resources:
   mybox:
    type: "OS::Nova::Server"
   properties:
       name: mybox
       image: { get_param: image }
      flavor: { get_param: flavor }
       key_name: { get_param: key_name }
```



And now we can



these parameters



```
heat stack-create -f stack.yml \
-P key_name=mykey
-P image=cirros-0.3.3-x86_64 \
mystack
```



How about we add some

NETWORK CONNECTIVITY

Wouldn't that be nice?



OS::Neutron::Net OS::Neutron::Subnet

Defines Neutron networks



```
mynet:
  type: "OS::Neutron::Net"
  properties:
    name: management-net
mysub_net:
  type: "OS::Neutron::Subnet"
  properties:
    name: management-sub-net
    network: { get_resource: management_net }
    cidr: 192.168.122.0/24
    gateway_ip: 192.168.101.1
    enable_dhcp: true
    allocation_pools:
      - start: "192.168.101.2"
        end: "192.168.101.50"
```



get_resource

Cross-reference between resources

Automatic dependency



OS::Neutron::RouterGateway OS::Neutron::RouterInterface

Configures Neutron routers



```
parameters:
 public_net:
   type: string
    description: Public network ID or name
resources:
 router:
   type: OS::Neutron::Router
 router_gateway:
   type: OS::Neutron::RouterGateway
    properties:
      router: { get_resource: router }
      network: { get_param: public_net }
 router_interface:
   type: OS::Neutron::RouterInterface
    properties:
      router: { get resource: router }
```



OS::Neutron::Port

Configures Neutron ports



```
mybox_management_port:
   type: "OS::Neutron::Port"
   properties:
    network: { get_resource: mynet }
```

```
mybox:
   type: "OS::Nova::Server"
   properties:
    name: deploy
   image: { get_param: image }
    flavor: { get_param: flavor }
    key_name: { get_param: key_name }
    networks:
    - port: { get_resource: mybox_management_port }
```



OS::Neutron::SecurityGroup

Configures Neutron security groups



```
mysecurity_group:
   type: OS::Neutron::SecurityGroup
properties:
   description: Neutron security group rules
   name: mysecurity_group
rules:
   - remote_ip_prefix: 0.0.0.0/0
   protocol: tcp
   port_range_min: 22
   port_range_max: 22
   - remote_ip_prefix: 0.0.0.0/0
   protocol: icmp
   direction: ingress
```

```
mybox_management_port:
    type: "OS::Neutron::Port"
    properties:
        network: { get_resource: mynet }
        security_groups:
        - { get_resource: mysecurity_group }
```



OS::Neutron::FloatingIP

Allocates floating IP addresses



```
myfloating_ip:
   type: "OS::Neutron::FloatingIP"
   properties:
     floating_network: { get_param: public_net }
     port: { get_resource: mybox_management_port }
```



outputs

Return stack values or attributes



```
outputs:
   public_ip:
    description: Floating IP address in public network
   value: { get_attr: [ myfloating_ip, floating_ip_address ] }
```



heat output-show \
mystack public_ip



Integrating



with

cloud-init



```
mybox:
  type: "OS::Nova::Server"
  properties:
    name: deploy
  image: { get_param: image }
    flavor: { get_param: flavor }
    key_name: { get_param: key_name }
    networks:
    - port: { get_resource: mybox_management_port }
    user_data: { get_file: cloud-config.yml }
    user_data_format: RAW
```



OS::Heat::CloudConfig

Manages cloud-config directly from Heat



```
resources:
   myconfig:
    type: "OS::Heat::CloudConfig"
   properties:
     cloud_config:
      package_update: true
     package_upgrade: true
```

```
mybox:
    type: "OS::Nova::Server"
    properties:
        name: deploy
    image: { get_param: image }
        flavor: { get_param: flavor }
        key_name: { get_param: key_name }
        networks:
            - port: { get_resource: mybox_management_port }
        user_data: { get_resource: myconfig }
        user_data_format: RAW
```



Now we can also



cloud-config PARAMETERS

directly from Heat



```
parameters:
    # [...]
    username:
    type: string
    description: Additional login username
    default: foobar
    gecos:
    type: string
    description: Additional user full name
    default: ''
```

```
myconfig:
  type: "OS::Heat::CloudConfig"
  properties:
    cloud_config:
      package_update: true
      package_upgrade: true
      users:

    default

      - name: { get_param: username }
        gecos: { get_param: gecos }
        groups: "users,adm"
        lock-passwd: false
        passwd: '$6$WP9924IJiLSto8Ng$MSDwCvlT28jM'
        shell: "/bin/bash"
        sudo: "ALL=(ALL) NOPASSWD:ALL"
      ssh_pwauth: true
```

CONGRATSI

You've just built a Heat/cloud-config enabled stack!



INFORMATION



http://cloudinit.readthedocs.org/

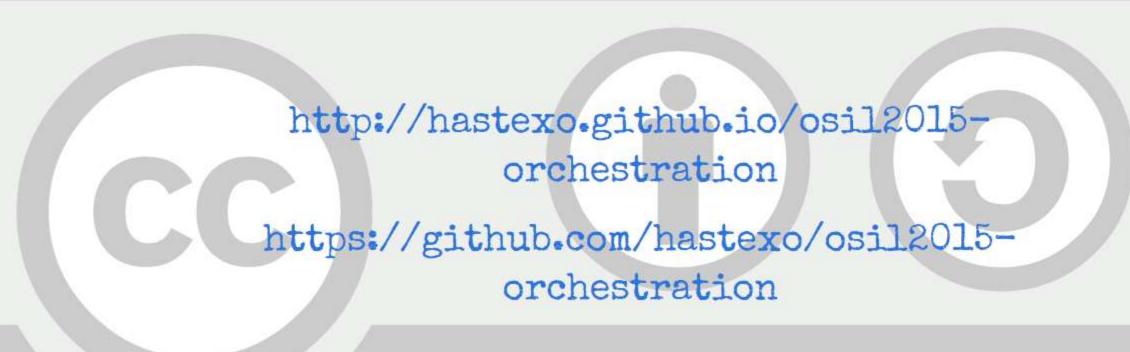


http://bazaar.launchpad.net/~cloud-initdev/cloudinit/trunk/files/head:/doc/examples/



http://docs.openstack.org/hot-reference/





BY SA



https://www.hastexo.com/openstack

