Cloud config examples

Including users and groups

```
#cloud-config
# Add groups to the system
# The following example adds the ubuntu group with members 'root' and 'sys'
# and the empty group cloud-users.
groups:
  - ubuntu: [root,sys]
  - cloud-users

# Add users to the system. Users are added after groups are added.
# Note: Most of these configuration options will not be honored if the user
#       already exists. Following options are the exceptions and they are
#       applicable on already-existing users:
#       - 'plain_text_passwd', 'hashed_passwd', 'lock_passwd', 'sudo',
#         'ssh_authorized_keys', 'ssh_redirect_user'.
users:
  - default
  - name: foobar
    gecos: Foo B. Bar
    primary_group: foobar
    groups: users
    selinux_user: staff_u
    expiredate: '2032-09-01'
    ssh_import_id: foobar
    lock_passwd: false
    passwd:
$6$j212wezy$7H/1LT4f9/N3wpgNunhsIqtMj62OKiS3nyNwuizouQc3u7MbYCarYeAHWYPYb2FT.lbioDm2RrkJ
  - name: barfoo
    gecos: Bar B. Foo
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: users, admin
    ssh_import_id: None
    lock_passwd: true
    ssh_authorized_keys:
      - <ssh pub key 1>
      - <ssh pub key 2>
  - name: cloudy
    gecos: Magic Cloud App Daemon User
    inactive: '5'
    system: true
  - name: fizzbuzz
    sudo: False
    ssh_authorized_keys:
      - <ssh pub key 1>
      - <ssh pub key 2>
  - snapuser: joe@joeuser.io
  - name: nosshlogins
    ssh_redirect_user: true

# Valid Values:
#   name: The user's login name
#   expiredate: Date on which the user's account will be disabled.
#   gecos: The user name's real name, i.e. "Bob B. Smith"
#   homedir: Optional. Set to the local path you want to use. Defaults to
```

```
#           /home/<username>
#   primary_group: define the primary group. Defaults to a new group created
#           named after the user.
#   groups:  Optional. Additional groups to add the user to. Defaults to none
#   selinux_user:  Optional. The SELinux user for the user's login, such as
#           "staff_u". When this is omitted the system will select the default
#           SELinux user.
#   lock_passwd: Defaults to true. Lock the password to disable password login
#   inactive: Number of days after password expires until account is disabled
#   passwd: The hash -- not the password itself -- of the password you want
#           to use for this user. You can generate a safe hash via:
#               mkpasswd --method=SHA-512 --rounds=4096
#           (the above command would create from stdin an SHA-512 password hash
#           with 4096 salt rounds)
#
#           Please note: while the use of a hashed password is better than
#               plain text, the use of this feature is not ideal. Also,
#               using a high number of salting rounds will help, but it should
#               not be relied upon.
#
#               To highlight this risk, running John the Ripper against the
#               example hash above, with a readily available wordlist, revealed
#               the true password in 12 seconds on a i7-2620QM.
#
#               In other words, this feature is a potential security risk and is
#               provided for your convenience only. If you do not fully trust the
#               medium over which your cloud-config will be transmitted, then you
#               should use SSH authentication only.
#
#               You have thus been warned.
#   no_create_home: When set to true, do not create home directory.
#   no_user_group: When set to true, do not create a group named after the user.
#   no_log_init: When set to true, do not initialize lastlog and faillog database.
#   ssh_import_id: Optional. Import SSH ids
#   ssh_authorized_keys: Optional. [list] Add keys to user's authorized keys file
#   ssh_redirect_user: Optional. [bool] Set true to block ssh logins for cloud
#       ssh public keys and emit a message redirecting logins to
#       use <default_username> instead. This option only disables cloud
#       provided public-keys. An error will be raised if ssh_authorized_keys
#       or ssh_import_id is provided for the same user.
#
#       ssh_authorized_keys.
#   sudo: Defaults to none. Accepts a sudo rule string, a list of sudo rule
#         strings or False to explicitly deny sudo usage. Examples:
#
#         Allow a user unrestricted sudo access.
#             sudo:  ALL=(ALL) NOPASSWD:ALL
#
#         Adding multiple sudo rule strings.
#             sudo:
#               - ALL=(ALL) NOPASSWD:/bin/mysql
#               - ALL=(ALL) ALL
#
#         Prevent sudo access for a user.
#             sudo: False
#
#         Note: Please double check your syntax and make sure it is valid.
#               cloud-init does not parse/check the syntax of the sudo
#               directive.
#   system: Create the user as a system user. This means no home directory.
#   snapuser: Create a Snappy (Ubuntu-Core) user via the snap create-user
#             command available on Ubuntu systems.  If the user has an account
#             on the Ubuntu SSO, specifying the email will allow snap to
#             request a username and any public ssh keys and will import
#             these into the system with username specifed by SSO account.
#             If 'username' is not set in SSO, then username will be the
#             shortname before the email domain.
```

```
#
#
# Default user creation:
#
# Unless you define users, you will get a 'ubuntu' user on ubuntu systems with the
# legacy permission (no password sudo, locked user, etc). If however, you want
# to have the 'ubuntu' user in addition to other users, you need to instruct
# cloud-init that you also want the default user. To do this use the following
# syntax:
#   users:
#     - default
#     - bob
#     - ....
#  foobar: ...
#
# users[0] (the first user in users) overrides the user directive.
#
# The 'default' user above references the distro's config:
# system_info:
#   default_user:
#     name: Ubuntu
#     plain_text_passwd: 'ubuntu'
#     home: /home/ubuntu
#     shell: /bin/bash
#     lock_passwd: True
#     gecos: Ubuntu
#     groups: [adm, audio, cdrom, dialout, floppy, video, plugdev, dip, netdev]
```

Writing out arbitrary files

```yaml
#cloud-config
# vim: syntax=yaml
#
# This is the configuration syntax that the write_files module
# will know how to understand. encoding can be given b64 or gzip or (gz+b64).
# The content will be decoded accordingly and then written to the path that is
# provided.
#
# Note: Content strings here are truncated for example purposes.
write_files:
- encoding: b64
  content: CiMgVGhpcyBmaWxlIGNvbnRyb2xzIHRoZSBzdGF0ZSBvZiBTRUxpbnV4...
  owner: root:root
  path: /etc/sysconfig/selinux
  permissions: '0644'
- content: |
    # My new /etc/sysconfig/samba file

    SMBDOPTIONS="-D"
  path: /etc/sysconfig/samba
- content: !!binary |
    f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAAwARAAAAAAABAAAAAAAAAAJAVAAAAAAAAAAAAAEAAOAAI
    AEAAHgAdAAYAAAAFAAAAQAAAAAAAAABAAEAAAAAAAEAAQAAAAAAAwAEAAAAAAADAAQAAAAAAAgA
    AAAAAAAAAwAAAAQAAAAAAgAAAAAAAAACQAAAAAAAAAJAAAAAAAAcAAAAAAAAABwAAAAAAAAAQAA
    ....
  path: /bin/arch
  permissions: '0555'
- encoding: gzip
  content: !!binary |
    H4sIAIDb/U8C/1NW1E/KzNMvzuBKTc7IV8hIzcnJVyjPL8pJ4QIA6N+MVxsAAAA=
  path: /usr/bin/hello
  permissions: '0755'
```

Adding a yum repository

```yaml
#cloud-config
# vim: syntax=yaml
#
# Add yum repository configuration to the system
#
# The following example adds the file /etc/yum.repos.d/epel_testing.repo
# which can then subsequently be used by yum for later operations.
yum_repos:
  # The name of the repository
  epel-testing:
    # Any repository configuration options
    # See: man yum.conf
    #
    # This one is required!
    baseurl: http://download.fedoraproject.org/pub/epel/testing/5/$basearch
    enabled: false
    failovermethod: priority
    gpgcheck: true
    gpgkey: file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL
    name: Extra Packages for Enterprise Linux 5 - Testing
```

Configure an instances trusted CA certificates

```yaml
#cloud-config
#
# This is an example file to configure an instance's trusted CA certificates
# system-wide for SSL/TLS trust establishment when the instance boots for the
# first time.
#
# Make sure that this file is valid yaml before starting instances.
# It should be passed as user-data when starting the instance.

ca-certs:
  # If present and set to True, the 'remove-defaults' parameter will remove
  # all the default trusted CA certificates that are normally shipped with
  # Ubuntu.
  # This is mainly for paranoid admins - most users will not need this
  # functionality.
  remove-defaults: true

  # If present, the 'trusted' parameter should contain a certificate (or list
  # of certificates) to add to the system as trusted CA certificates.
  # Pay close attention to the YAML multiline list syntax.  The example shown
  # here is for a list of multiline certificates.
  trusted:
  - |
   -----BEGIN CERTIFICATE-----
   YOUR-ORGS-TRUSTED-CA-CERT-HERE
   -----END CERTIFICATE-----
  - |
   -----BEGIN CERTIFICATE-----
   YOUR-ORGS-TRUSTED-CA-CERT-HERE
   -----END CERTIFICATE-----
```

Configure an instances resolv.conf

Note: when using a config drive and a RHEL like system resolv.conf will also be managed 'automatically' due to the available information provided for dns servers in the config drive network format. For those that wish to have different settings use this module.

```yaml
#cloud-config
#
# This is an example file to automatically configure resolv.conf when the
# instance boots for the first time.
#
# Ensure that your yaml is valid and pass this as user-data when starting
# the instance. Also be sure that your cloud.cfg file includes this
# configuration module in the appropriate section.
#
manage_resolv_conf: true

resolv_conf:
  nameservers: ['8.8.4.4', '8.8.8.8']
  searchdomains:
    - foo.example.com
    - bar.example.com
  domain: example.com
  options:
    rotate: true
    timeout: 1
```

Install and run chef recipes

```
#cloud-config
#
# This is an example file to automatically install chef-client and run a
# list of recipes when the instance boots for the first time.
# Make sure that this file is valid yaml before starting instances.
# It should be passed as user-data when starting the instance.
#
# This example assumes the instance is 16.04 (xenial)


# The default is to install from packages.

# Key from https://packages.chef.io/chef.asc
apt:
  sources:
    source1:
      source: "deb http://packages.chef.io/repos/apt/stable $RELEASE main"
      key: |
        -----BEGIN PGP PUBLIC KEY BLOCK-----
        Version: GnuPG v1.4.12 (Darwin)
        Comment: GPGTools - http://gpgtools.org

        mQGiBEppC7QRBADfsOkZU6KZK+YmKw4wev5mjKJEkVGlus+NxW8wItX5sGa6kdUu
        twAyj7Yr92rF+ICFEP3gGU6+lGo0Nve7KxkN/1W7/m3G4zuk+ccIKmjp8KS3qn99
        dxy64vcji9jIllVa+XXOGIp0G8GEaj7mbkixL/bMeGfdMlv8Gf2XPpp9vwCgn/GC
        JKacfnw7MpLKUHOYSlb//JsEAJqao3ViNfav83jJKEkD8cf59Y8xKia5OpZqTK5W
        ShVnNWS3U5IVQk10ZDH97Qn/YrK387H4CyhLE9mxPXs/ul18ioiaars/q2MEKU2I
        XKfV21eMLO9LYd6Ny/Kqj8o5WQK2J6+NAhSwvthZcIEphcFignIuobP+B5wNFQpe
        DbKfA/0WvN2OwFeWRcmmd3Hz7nHTpcnSF+4QX6yHRF/5BgxkG6IqBIACQbzPn6Hm
        sMtm/SVf11izmDqSsQptCrOZILfLX/mE+YOl+CwWSHhl+YsFts1WOuh1EhQD26aO
        Z84HuHV5HFRWjDLw9LriltBVQcXbpfSrRP5bdr7Wh8vhqJTPjrQnT3BzY29kZSBQ
        YWNrYWdlcyA8cGFja2FnZXNAb3BzY29kZS5jb20+iGAEExECACAFAkppC7QCGwMG
        CwkIBwMCBBUCCAMEFgIDAQIeAQIXgAAKCRApQKupg++Caj8sAKCOXmdG36gWji/K
        +o+XtBfvdMnFYQCfTCEWxRy2BnzLoBBFCjDSK6sJqCu0IENIRUYgUGFja2FnZXMg
        PHBhY2thZ2VzQGNoZWYuaW8+iGIEExECACIFAlQwYFECGwMGCwkIBwMCBhUIAgkK
        CwQWAgMBAh4BAheAAAoJECl Aq6mD74JqX94An26z99XOHWpLN8ahzm7cp13t4Xid
        AJ9wVcgoUBzvgg91lKfv/34cmemZn7kCDQRKaQu0EAgAg7ZLCVGVTmLqBM6njZEd
        Zbv+mZbvwLBSomdiqddE6u3eH0X3GuwaQfQWHUVG2yedyDMiG+EMtCdEeeRebTCz
        SNXQ8Xvi22hRPoEsBSwWLZI8/XNg0n0f1+GEr+mOKO0BxDB2DG7DA0nnEISxwFkK
        OFJFebR3fRsrWjj0KjDxkhse2ddU/jVz1BY7Nf8toZmwpBmdozETMOTx3LJy1HZ/
        Te9FJXJMUaB2lRyluv15MVWCKQJro4MQG/7QGcIfrIZNfAGJ32DDSjV7/YO+IpRY
        IL4CUBQ65suY4gYUG4jhRH6u7H1p99sdwsg5OIpBe/v2Vbc/tbwAB+eJJAp89Zeu
        twADBQf/ZcGoPhTGFuzbkcNRSIz+boaeWPoSxK2DyfScyCAuG41CY9+g0HIw9Sq8
        DuxQvJ+vrEJjNvNE3EAEdKl/zkXMZDb1EXjGwDi845TxEMhhD1dDw2qpHqnJ2mtE
        WpZ7juGwA3sGhi6FapO04tIGacCfNNHmlRGipyq5ZiKIRq9mLEndlECr8cwaKgkS
        0wWu+xmMZe7N5/t/TK19HXNh4tVacv0F3fYK54GUjt2FjCQV75USnmNY4KPTYLXA
        dzC364hEMlXpN21siIFgB04w+TXn5UF3B4FfAy5hevvr4DtV4MvMiGLu0oWjpaLC
        MpmrR3Ny2wkmO0h+vgri9uIP06ODWIhJBBgRAgAJBQJKaQu0AhsMAAoJECl Aq6mD
        74Jq4hIAoJ5KrYS8kCwj26SAGzglwggpvt3CAJ0bekyky56vNqoegB+y4PQVDv4K
        zA==
        =IxPr
        -----END PGP PUBLIC KEY BLOCK-----

chef:

  # Valid values are 'accept' and 'accept-no-persist'
  chef_license: "accept"

  # Valid values are 'gems' and 'packages' and 'omnibus'
  install_type: "packages"

  # Boolean: run 'install_type' code even if chef-client
  #          appears already installed.
```

```
    force_install: false

    # Chef settings
    server_url: "https://chef.yourorg.com"

    # Node Name
    # Defaults to the instance-id if not present
    node_name: "your-node-name"

    # Environment
    # Defaults to '_default' if not present
    environment: "production"

    # Default validation name is chef-validator
    validation_name: "yourorg-validator"
    # if validation_cert's value is "system" then it is expected
    # that the file already exists on the system.
    validation_cert: |
      -----BEGIN RSA PRIVATE KEY-----
      YOUR-ORGS-VALIDATION-KEY-HERE
      -----END RSA PRIVATE KEY-----

    # A run list for a first boot json, an example (not required)
    run_list:
      - "recipe[apache2]"
      - "role[db]"

    # Specify a list of initial attributes used by the cookbooks
    initial_attributes:
      apache:
        prefork:
          maxclients: 100
        keepalive: "off"

    # if install_type is 'omnibus', change the url to download
    omnibus_url: "https://www.chef.io/chef/install.sh"

    # if install_type is 'omnibus', pass pinned version string
    # to the install script
    omnibus_version: "12.3.0"

    # If encrypted data bags are used, the client needs to have a secrets file
    # configured to decrypt them
    encrypted_data_bag_secret: "/etc/chef/encrypted_data_bag_secret"

  # Capture all subprocess output into a logfile
  # Useful for troubleshooting cloud-init issues
  output: {all: '| tee -a /var/log/cloud-init-output.log'}
```

Setup and run puppet

```
#cloud-config
#
# This is an example file to automatically setup and run puppetd
# when the instance boots for the first time.
# Make sure that this file is valid yaml before starting instances.
# It should be passed as user-data when starting the instance.
puppet:
  # Every key present in the conf object will be added to puppet.conf:
  # [name]
  # subkey=value
  #
  # For example the configuration below will have the following section
  # added to puppet.conf:
  # [puppetd]
  # server=puppetmaster.example.org
  # certname=i-0123456.ip-X-Y-Z.cloud.internal
  #
  # The puppmaster ca certificate will be available in
  # /var/lib/puppet/ssl/certs/ca.pem
  conf:
    agent:
      server: "puppetmaster.example.org"
      # certname supports substitutions at runtime:
      #   %i: instanceid
      #       Example: i-0123456
      #   %f: fqdn of the machine
      #       Example: ip-X-Y-Z.cloud.internal
      #
      # NB: the certname will automatically be lowercased as required by puppet
      certname: "%i.%f"
    # ca_cert is a special case. It won't be added to puppet.conf.
    # It holds the puppetmaster certificate in pem format.
    # It should be a multi-line string (using the | yaml notation for
    # multi-line strings).
    # The puppetmaster certificate is located in
    # /var/lib/puppet/ssl/ca/ca_crt.pem on the puppetmaster host.
    #
    ca_cert: |
      -----BEGIN CERTIFICATE-----
      MIICCTCCAXKgAwIBAgIBATANBgkqhkiG9w0BAQUFADANMQswCQYDVQQDDAJjYTAe
      Fw0xMDAyMTUxNzI5MjFaFw0xNTAyMTQxNzI5MjFaMA0xCzAJBgNVBAMMAmNhMIGf
      MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCu7Q40sm47/E1Pf+r8AYb/V/FWGPgc
      b014OmNoX7dgCxTDvps/h8Vw555PdAFsW5+QhsGr31IJNI3kSYprFQcYf7A8tNWu
      1MASW2CfaEiOEi9F1R3R4Qlz4ix+iNoHiUDTjazw/tZwEdxaQXQVLwgTGRwVa+aA
      qbutJKi93MILLwIDAQABo3kwdzA4BglghkgBhvhCAQ0EKxYpUHVwcGV0IFJ1Ynkv
      T3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwDwYDVR0TAQH/BAUwAwEB/zAd
      BgNVHQ4EFgQUu4+jHB+GYE5Vxo+ol1OAhevspjAwCwYDVR0PBAQDAgEGMA0GCSqG
      SIb3DQEBBQUAA4GBAH/rxlUIjwNb3n7TXJcDJ6MMHUlwjr03BDJXKb34Ulndkpaf
      +GAlzPXWa7bO908M9I8RnPfvtKnteLbvgTK+h+zX1XCty+S2EQWk29i2AdoqOTxb
      hppiGMp0tT5Havu4aceCXiy2crVcudj3NFciy8X66SoECemW9UYDCb9T5D0d
      -----END CERTIFICATE-----
```

Add primary apt repositories

```
#cloud-config

# Add primary apt repositories
#
# To add 3rd party repositories, see cloud-config-apt.txt or the
# Additional apt configuration and repositories section.
#
#
# Default: auto select based on cloud metadata
#  in ec2, the default is <region>.archive.ubuntu.com
# apt:
#   primary:
#     - arches [default]
#       uri:
#     use the provided mirror
#       search:
#     search the list for the first mirror.
#     this is currently very limited, only verifying that
#     the mirror is dns resolvable or an IP address
#
# if neither mirror is set (the default)
# then use the mirror provided by the DataSource found.
# In EC2, that means using <region>.ec2.archive.ubuntu.com
#
# if no mirror is provided by the DataSource, but 'search_dns' is
# true, then search for dns names '<distro>-mirror' in each of
# - fqdn of this host per cloud metadata
# - localdomain
# - no domain (which would search domains listed in /etc/resolv.conf)
# If there is a dns entry for <distro>-mirror, then it is assumed that there
# is a distro mirror at http://<distro>-mirror.<domain>/<distro>
#
# That gives the cloud provider the opportunity to set mirrors of a distro
# up and expose them only by creating dns entries.
#
# if none of that is found, then the default distro mirror is used
apt:
  primary:
    - arches: [default]
      uri: http://us.archive.ubuntu.com/ubuntu/
# or
apt:
  primary:
    - arches: [default]
      search:
        - http://local-mirror.mydomain
        - http://archive.ubuntu.com
# or
apt:
  primary:
    - arches: [default]
      search_dns: True
```

Run commands on first boot

```
#cloud-config

# boot commands
# default: none
# this is very similar to runcmd, but commands run very early
# in the boot process, only slightly after a 'boothook' would run.
# bootcmd should really only be used for things that could not be
# done later in the boot process.  bootcmd is very much like
# boothook, but possibly with more friendly.
# - bootcmd will run on every boot
# - the INSTANCE_ID variable will be set to the current instance id.
# - you can use 'cloud-init-per' command to help only run once
bootcmd:
  - echo 192.168.1.130 us.archive.ubuntu.com >> /etc/hosts
  - [ cloud-init-per, once, mymkfs, mkfs, /dev/vdb ]
```

```
#cloud-config

# run commands
# default: none
# runcmd contains a list of either lists or a string
# each item will be executed in order at rc.local like level with
# output to the console
# - runcmd only runs during the first boot
# - if the item is a list, the items will be properly executed as if
#   passed to execve(3) (with the first arg as the command).
# - if the item is a string, it will be simply written to the file and
#   will be interpreted by 'sh'
#
# Note, that the list has to be proper yaml, so you have to quote
# any characters yaml would eat (':' can be problematic)
runcmd:
 - [ ls, -l, / ]
 - [ sh, -xc, "echo $(date) ': hello world!'" ]
 - [ sh, -c, echo "=========hello world'=========" ]
 - ls -l /root
 # Note: Don't write files to /tmp from cloud-init use /run/somedir instead.
 # Early boot environments can race systemd-tmpfiles-clean LP: #1707222.
 - mkdir /run/mydir
 - [ wget, "http://slashdot.org", -O, /run/mydir/index.html ]
```

Alter the completion message

```
#cloud-config

# final_message
# default: cloud-init boot finished at $TIMESTAMP. Up $UPTIME seconds
# this message is written by cloud-final when the system is finished
# its first boot
final_message: "The system is finally up, after $UPTIME seconds"
```

Install arbitrary packages

```
#cloud-config

# Install additional packages on first boot
#
# Default: none
#
# if packages are specified, this apt_update will be set to true
#
# packages may be supplied as a single package name or as a list
# with the format [<package>, <version>] wherein the specifc
# package version will be installed.
packages:
 - pwgen
 - pastebinit
 - [libpython2.7, 2.7.3-0ubuntu3.1]
```

Update apt database on first boot

```
#cloud-config
# Update apt database on first boot (run 'apt-get update').
# Note, if packages are given, or package_upgrade is true, then
# update will be done independent of this setting.
#
# Default: false
# Aliases: apt_update
package_update: true
```

Run apt or yum upgrade

```
#cloud-config

# Upgrade the instance on first boot
# (ie run apt-get upgrade)
#
# Default: false
# Aliases: apt_upgrade
package_upgrade: true
```

Adjust mount points mounted

```
#cloud-config

# set up mount points
# 'mounts' contains a list of lists
#  the inner list are entries for an /etc/fstab line
#  ie : [ fs_spec, fs_file, fs_vfstype, fs_mntops, fs-freq, fs_passno ]
#
# default:
# mounts:
#  - [ ephemeral0, /mnt ]
#  - [ swap, none, swap, sw, 0, 0 ]
#
# in order to remove a previously listed mount (ie, one from defaults)
# list only the fs_spec.  For example, to override the default, of
# mounting swap:
# - [ swap ]
# or
# - [ swap, null ]
#
# - if a device does not exist at the time, an entry will still be
#   written to /etc/fstab.
# - '/dev' can be ommitted for device names that begin with: xvd, sd, hd, vd
# - if an entry does not have all 6 fields, they will be filled in
#   with values from 'mount_default_fields' below.
#
# Note, that you should set 'nofail' (see man fstab) for volumes that may not
# be attached at instance boot (or reboot).
#
mounts:
 - [ ephemeral0, /mnt, auto, "defaults,noexec" ]
 - [ sdc, /opt/data ]
 - [ xvdh, /opt/data, "auto", "defaults,nofail", "0", "0" ]
 - [ dd, /dev/zero ]

# mount_default_fields
# These values are used to fill in any entries in 'mounts' that are not
# complete.  This must be an array, and must have 6 fields.
mount_default_fields: [ None, None, "auto", "defaults,nofail", "0", "2" ]


# swap can also be set up by the 'mounts' module
# default is to not create any swap files, because 'size' is set to 0
swap:
  filename: /swap.img
  size: "auto" # or size in bytes
  maxsize: size in bytes
```

Call a url when finished

```yaml
#cloud-config

# phone_home: if this dictionary is present, then the phone_home
# cloud-config module will post specified data back to the given
# url
# default: none
# phone_home:
#   url: http://my.foo.bar/$INSTANCE/
#   post: all
#   tries: 10
#
phone_home:
  url: http://my.example.com/$INSTANCE_ID/
  post: [ pub_key_dsa, pub_key_rsa, pub_key_ecdsa, instance_id ]
```

Reboot/poweroff when finished

```yaml
#cloud-config

## poweroff or reboot system after finished
# default: none
#
# power_state can be used to make the system shutdown, reboot or
# halt after boot is finished.  This same thing can be acheived by
# user-data scripts or by runcmd by simply invoking 'shutdown'.
#
# Doing it this way ensures that cloud-init is entirely finished with
# modules that would be executed, and avoids any error/log messages
# that may go to the console as a result of system services like
# syslog being taken down while cloud-init is running.
#
# If you delay '+5' (5 minutes) and have a timeout of
# 120 (2 minutes), then the max time until shutdown will be 7 minutes.
# cloud-init will invoke 'shutdown +5' after the process finishes, or
# when 'timeout' seconds have elapsed.
#
# delay: form accepted by shutdown.  default is 'now'. other format
#        accepted is '+m' (m in minutes)
# mode: required. must be one of 'poweroff', 'halt', 'reboot'
# message: provided as the message argument to 'shutdown'. default is none.
# timeout: the amount of time to give the cloud-init process to finish
#          before executing shutdown.
# condition: apply state change only if condition is met.
#            May be boolean True (always met), or False (never met),
#            or a command string or list to be executed.
#            command's exit code indicates:
#               0: condition met
#               1: condition not met
#            other exit codes will result in 'not met', but are reserved
#            for future use.
#
power_state:
  delay: "+30"
  mode: poweroff
  message: Bye Bye
  timeout: 30
  condition: True
```

Configure instances SSH keys

```
#cloud-config

# add each entry to ~/.ssh/authorized_keys for the configured user or the
# first user defined in the user definition directive.
ssh_authorized_keys:
  - ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAGEA3FSyQwBI6Z+nCSjUUk8EEAnnkhXlukKoUPND/RRClWz2s5TCz
  - ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEA3I7VUf2l5gSn5uavROsc5HRDpZdQueUq5ozemNSj8T7enqKHO

# Send pre-generated SSH private keys to the server
# If these are present, they will be written to /etc/ssh and
# new random keys will not be generated
#  in addition to 'rsa' and 'dsa' as shown below, 'ecdsa' is also supported
ssh_keys:
  rsa_private: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIBxwIBAAJhAKD0YSHy73nUgysO13XsJmd4fHiFyQ+00R7VVu2iV9Qcon2LZS/x
    1cydPZ4pQpfjEha6WxZ6o8ci/Ea/w0n+0HGPwaxlEG2Z9inNtj3pgFrYcRztfECb
    1j6HCibZbAzYtwIBIwJgO8h72WjcmvcpZ8OvHSvTwAguO2TkR6mPgHsgSaKy6GJo
    PUJnaZRWuba/HX0KGyhz19nPzLpzG5f0fYahlMJAyc13FV7K6kMBPXTRR6FxgHEg
    L0MPC7cdqAwOVNcPY6A7AjEA1bNaIjOzFN2sfZX0j7OMhQuc4zP7r80zaGc5oy6W
    p58hRAncFKEvnEq2CeL3vtuZAjEAwNBHpbNsBYTRPCHM7rZuG/iBtwp8Rxhc9I5w
    ixvzMgi+HpGLWzUIBS+P/XhekIjPAjA285rVmEP+DR255Ls65QbgYhJmTzIXQ2T9
    luLvcmFBC6l35Uc4gTgg4ALsmXLn71MCMGMpSWspEvuGInayTCL+vEjmNBT+FAdO
    W7D4zCpI43jRS9U06JVOeSc9CDk2lwiA3wIwCTB/6uc8Cq85D9YqpM10FuHjKpnP
    REPPOyrAspdeOAV+6VKRavstea7+2DZmSUgE
    -----END RSA PRIVATE KEY-----

  rsa_public: ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAGEAoPRhIfLvedSDKw7XdewmZ3h8eIXJD7TRHtVW7aJ

  dsa_private: |
    -----BEGIN DSA PRIVATE KEY-----
    MIIBuwIBAAKBgQDP2HLu7pTExL89USyM0264RCyWX/CMLmukxX0Jdbm29ax8FBJT
    pLrO8TIXVY5rPAJm1dTHnpuyJhOvU9G7M8tPUABtzSJh4GVSHlwaCfycwcpLv9TX
    DgWIpSj+6EiHCyaRlB1/CBp9RiaB+10QcFbm+lapuET+/Au6vSDp9IRtlQIVAIMR
    8KucvUYbOEI+yv+5LW9u3z/BAoGBAI0q6JP+JvJmwZFaeCMMVxXUbqiSko/P1lsa
    LNNBHZ5/8MOUIm8rB2FC6ziidfueJpqTMqeQmSAlEBCwnwreUnGfRrKoJpyPNENY
    d15MG6N5J+z81sEcHFeprryZ+D3Ge9VjPq3Tf3NhKKwCDQ0240aPezbnjPeFm4mH
    bYxxcZ9GAoGAXmLIFSQgiAPu459rCKxT46tHJtM0QfnNiEnQLbFluefZ/yiI4DI3
    8UzTCOXLhUA7ybmZha+D/csj15Y9/BNFuO7unzVhikCQV9DTeXX46pG4s1o23JKC
    /QaYWNMZ7kTRv+wWow9MhGiVdML4ZN4XnifuO5krqAybngIy66PMEoQCFEIsKKWv
    99iziAH0KBMVbxy03Trz
    -----END DSA PRIVATE KEY-----

  dsa_public: ssh-dss
AAAAB3NzaC1kc3MAAACBAM/Ycu7ulMTEvz1RLIzTbrhELJZf8Iwua6TFfQl1ubb1rHwUElOkus7xMhdVjms8AmbV1
smoser@localhost

# By default, the fingerprints of the authorized keys for the users
# cloud-init adds are printed to the console. Setting
# no_ssh_fingerprints to true suppresses this output.
no_ssh_fingerprints: false

# By default, (most) ssh host keys are printed to the console. Setting
# emit_keys_to_console to false suppresses this output.
ssh:
  emit_keys_to_console: false
```

Additional apt configuration and repositories

```yaml
#cloud-config
# apt_pipelining (configure Acquire::http::Pipeline-Depth)
# Default: disables HTTP pipelining. Certain web servers, such
# as S3 do not pipeline properly (LP: #948461).
# Valid options:
#   False/default: Disables pipelining for APT
#   None/Unchanged: Use OS default
#   Number: Set pipelining to some number (not recommended)
apt_pipelining: False

## apt config via system_info:
# under the 'system_info', you can customize cloud-init's interaction
# with apt.
#  system_info:
#    apt_get_command: [command, argument, argument]
#    apt_get_upgrade_subcommand: dist-upgrade
#
# apt_get_command:
#  To specify a different 'apt-get' command, set 'apt_get_command'.
#  This must be a list, and the subcommand (update, upgrade) is appended to it.
#  default is:
#    ['apt-get', '--option=Dpkg::Options::=--force-confold',
#     '--option=Dpkg::options::=--force-unsafe-io', '--assume-yes', '--quiet']
#
# apt_get_upgrade_subcommand: "dist-upgrade"
#  Specify a different subcommand for 'upgrade. The default is 'dist-upgrade'.
#  This is the subcommand that is invoked for package_upgrade.
#
# apt_get_wrapper:
#   command: eatmydata
#   enabled: [True, False, "auto"]
#

# Install additional packages on first boot
#
# Default: none
#
# if packages are specified, this apt_update will be set to true

packages: ['pastebinit']

apt:
  # The apt config consists of two major "areas".
  #
  # On one hand there is the global configuration for the apt feature.
  #
  # On one hand (down in this file) there is the source dictionary which allows
  # to define various entries to be considered by apt.

  ################################################################################
  # Section 1: global apt configuration
  #
  # The following examples number the top keys to ease identification in
  # discussions.

  # 1.1 preserve_sources_list
  #
  # Preserves the existing /etc/apt/sources.list
  # Default: false - do overwrite sources_list. If set to true then any
  # "mirrors" configuration will have no effect.
  # Set to true to avoid affecting sources.list. In that case only
  # "extra" source specifications will be written into
  # /etc/apt/sources.list.d/*
  preserve_sources_list: true

  # 1.2 disable_suites
  #
```

```
    # This is an empty list by default, so nothing is disabled.
    #
    # If given, those suites are removed from sources.list after all other
    # modifications have been made.
    # Suites are even disabled if no other modification was made,
    # but not if is preserve_sources_list is active.
    # There is a special alias "$RELEASE" as in the sources that will be replace
    # by the matching release.
    #
    # To ease configuration and improve readability the following common ubuntu
    # suites will be automatically mapped to their full definition.
    # updates   => $RELEASE-updates
    # backports => $RELEASE-backports
    # security  => $RELEASE-security
    # proposed  => $RELEASE-proposed
    # release   => $RELEASE
    #
    # There is no harm in specifying a suite to be disabled that is not found in
    # the source.list file (just a no-op then)
    #
    # Note: Lines don't get deleted, but disabled by being converted to a comment.
    # The following example disables all usual defaults except $RELEASE-security.
    # On top it disables a custom suite called "mysuite"
    disable_suites: [$RELEASE-updates, backports, $RELEASE, mysuite]

    # 1.3 primary/security archives
    #
    # Default: none - instead it is auto select based on cloud metadata
    # so if neither "uri" nor "search", nor "search_dns" is set (the default)
    # then use the mirror provided by the DataSource found.
    # In EC2, that means using <region>.ec2.archive.ubuntu.com
    #
    # define a custom (e.g. localized) mirror that will be used in sources.list
    # and any custom sources entries for deb / deb-src lines.
    #
    # One can set primary and security mirror to different uri's
    # the child elements to the keys primary and secondary are equivalent
    primary:
      # arches is list of architectures the following config applies to
      # the special keyword "default" applies to any architecture not explicitly
      # listed.
      - arches: [amd64, i386, default]
        # uri is just defining the target as-is
        uri: http://us.archive.ubuntu.com/ubuntu
        #
        # via search one can define lists that are tried one by one.
        # The first with a working DNS resolution (or if it is an IP) will be
        # picked. That way one can keep one configuration for multiple
        # subenvironments that select the working one.
        search:
          - http://cool.but-sometimes-unreachable.com/ubuntu
          - http://us.archive.ubuntu.com/ubuntu
        # if no mirror is provided by uri or search but 'search_dns' is
        # true, then search for dns names '<distro>-mirror' in each of
        # - fqdn of this host per cloud metadata
        # - localdomain
        # - no domain (which would search domains listed in /etc/resolv.conf)
        # If there is a dns entry for <distro>-mirror, then it is assumed that
        # there is a distro mirror at http://<distro>-mirror.<domain>/<distro>
        #
        # That gives the cloud provider the opportunity to set mirrors of a distro
        # up and expose them only by creating dns entries.
        #
        # if none of that is found, then the default distro mirror is used
        search_dns: true
        #
        # If multiple of a category are given
```

```
      #   1. uri
      #   2. search
      #   3. search_dns
      # the first defining a valid mirror wins (in the order as defined here,
      # not the order as listed in the config).
      #
      # Additionally, if the repository requires a custom signing key, it can be
      # specified via the same fields as for custom sources:
      #   'keyid': providing a key to import via shortid or fingerprint
      #   'key': providing a raw PGP key
      #   'keyserver': specify an alternate keyserver to pull keys from that
      #                were specified by keyid
    - arches: [s390x, arm64]
      # as above, allowing to have one config for different per arch mirrors
  # security is optional, if not defined it is set to the same value as primary
  security:
    - uri: http://security.ubuntu.com/ubuntu
  # If search_dns is set for security the searched pattern is:
  #   <distro>-security-mirror

  # if no mirrors are specified at all, or all lookups fail it will try
  # to get them from the cloud datasource and if those neither provide one fall
  # back to:
  #   primary: http://archive.ubuntu.com/ubuntu
  #   security: http://security.ubuntu.com/ubuntu

  # 1.4 sources_list
  #
  # Provide a custom template for rendering sources.list
  # without one provided cloud-init uses builtin templates for
  # ubuntu and debian.
  # Within these sources.list templates you can use the following replacement
  # variables (all have sane Ubuntu defaults, but mirrors can be overwritten
  # as needed (see above)):
  # => $RELEASE, $MIRROR, $PRIMARY, $SECURITY
  sources_list: | # written by cloud-init custom template
    deb $MIRROR $RELEASE main restricted
    deb-src $MIRROR $RELEASE main restricted
    deb $PRIMARY $RELEASE universe restricted
    deb $SECURITY $RELEASE-security multiverse

  # 1.5 conf
  #
  # Any apt config string that will be made available to apt
  # see the APT.CONF(5) man page for details what can be specified
  conf: | # APT config
    APT {
      Get {
        Assume-Yes "true";
        Fix-Broken "true";
      };
    };

  # 1.6 (http_|ftp_|https_)proxy
  #
  # Proxies are the most common apt.conf option, so that for simplified use
  # there is a shortcut for those. Those get automatically translated into the
  # correct Acquire::*::Proxy statements.
  #
  # note: proxy actually being a short synonym to http_proxy
  proxy: http://[[user][:pass]@]host[:port]/
  http_proxy: http://[[user][:pass]@]host[:port]/
  ftp_proxy: ftp://[[user][:pass]@]host[:port]/
  https_proxy: https://[[user][:pass]@]host[:port]/

  # 1.7 add_apt_repo_match
  #
```

```
  # 'source' entries in apt-sources that match this python regex
  # expression will be passed to add-apt-repository
  # The following example is also the builtin default if nothing is specified
  add_apt_repo_match: '^[\w-]+:\w'


  ############################################################################
  # Section 2: source list entries
  #
  # This is a dictionary (unlike most block/net which are lists)
  #
  # The key of each source entry is the filename and will be prepended by
  # /etc/apt/sources.list.d/ if it doesn't start with a '/'.
  # If it doesn't end with .list it will be appended so that apt picks up it's
  # configuration.
  #
  # Whenever there is no content to be written into such a file, the key is
  # not used as filename - yet it can still be used as index for merging
  # configuration.
  #
  # The values inside the entries consost of the following optional entries:
  #   'source': a sources.list entry (some variable replacements apply)
  #   'keyid': providing a key to import via shortid or fingerprint
  #   'key': providing a raw PGP key
  #   'keyserver': specify an alternate keyserver to pull keys from that
  #                were specified by keyid

  # This allows merging between multiple input files than a list like:
  # cloud-config1
  # sources:
  #   s1: {'key': 'key1', 'source': 'source1'}
  # cloud-config2
  # sources:
  #   s2: {'key': 'key2'}
  #   s1: {'keyserver': 'foo'}
  # This would be merged to
  # sources:
  #   s1:
  #     keyserver: foo
  #     key: key1
  #     source: source1
  #   s2:
  #     key: key2
  #
  # The following examples number the subfeatures per sources entry to ease
  # identification in discussions.


  sources:
    curtin-dev-ppa.list:
      # 2.1 source
      #
      # Creates a file in /etc/apt/sources.list.d/ for the sources list entry
      # based on the key: "/etc/apt/sources.list.d/curtin-dev-ppa.list"
      source: "deb http://ppa.launchpad.net/curtin-dev/test-archive/ubuntu xenial
main"

      # 2.2 keyid
      #
      # Importing a gpg key for a given key id. Used keyserver defaults to
      # keyserver.ubuntu.com
      keyid: F430BBA5 # GPG key ID published on a key server

    ignored1:
      # 2.3 PPA shortcut
      #
      # Setup correct apt sources.list line and Auto-Import the signing key
```

```
      # from LP
      #
      # See https://help.launchpad.net/Packaging/PPA for more information
      # this requires 'add-apt-repository'. This will create a file in
      # /etc/apt/sources.list.d automatically, therefore the key here is
      # ignored as filename in those cases.
      source: "ppa:curtin-dev/test-archive"    # Quote the string

    my-repo2.list:
      # 2.4 replacement variables
      #
      # sources can use $MIRROR, $PRIMARY, $SECURITY and $RELEASE replacement
      # variables.
      # They will be replaced with the default or specified mirrors and the
      # running release.
      # The entry below would be possibly turned into:
      #   source: deb http://archive.ubuntu.com/ubuntu xenial multiverse
      source: deb $MIRROR $RELEASE multiverse

    my-repo3.list:
      # this would have the same end effect as 'ppa:curtin-dev/test-archive'
      source: "deb http://ppa.launchpad.net/curtin-dev/test-archive/ubuntu xenial
main"
      keyid: F430BBA5 # GPG key ID published on the key server
      filename: curtin-dev-ppa.list

    ignored2:
      # 2.5 key only
      #
      # this would only import the key without adding a ppa or other source spec
      # since this doesn't generate a source.list file the filename key is ignored
      keyid: F430BBA5 # GPG key ID published on a key server

    ignored3:
      # 2.6 key id alternatives
      #
      # Keyid's can also be specified via their long fingerprints
      keyid: B59D 5F15 97A5 04B7 E230  6DCA 0620 BBCF 0368 3F77

    ignored4:
      # 2.7 alternative keyservers
      #
      # One can also specify alternative keyservers to fetch keys from.
      keyid: B59D 5F15 97A5 04B7 E230  6DCA 0620 BBCF 0368 3F77
      keyserver: pgp.mit.edu


    my-repo4.list:
      # 2.8 raw key
      #
      # The apt signing key can also be specified by providing a pgp public key
      # block. Providing the PGP key this way is the most robust method for
      # specifying a key, as it removes dependency on a remote key server.
      #
      # As with keyid's this can be specified with or without some actual source
      # content.
      key: | # The value needs to start with -----BEGIN PGP PUBLIC KEY BLOCK-----
        -----BEGIN PGP PUBLIC KEY BLOCK-----
        Version: SKS 1.0.10

        mI0ESpA3UQEEALdZKVIMq0j6qWAXAyxSlF63SvPVIgxHPb9Nk0DZUixn+akqytxG4zKCONz6
        qLjoBBfHnynyVLfT4ihg9an1PqxRnTO+JKQxl8NgKGz6Pon569GtAOdWNKw15XKinJTDLjnj
        9y96ljJqRcpV9t/WsIcdJPcKFR5voHTEoABE2aEXABEBAAG0GUxhdW5jaHBhZCBQUEEgZm9y
        IEFsZXN0aWOItgQTAQIAIAUCSpA3UQIbAwYLCQgHAwIEFQIIAwQWAgMBAh4BAheAAAoJEA7H
        5Qi+CcVxWZ8D/1MyYvfj3FJPZUm2Yo1zZsQ657vHI9+pPouqflWOayRR9jbiyUFIn0VdQBrP
        t0FwvnOFArUovUWoKAEdqR8hPy3M3APUZjl5K4cMZR/xaMQeQRZ5CHpS4DBKURKAHC0ltS5o
        uBJKQOZm5iltJp15cgyIkBkGe8Mx18VFyVglAZey
```

```
=Y2oI
-----END PGP PUBLIC KEY BLOCK-----
```

Disk setup

```
#cloud-config
# Cloud-init supports the creation of simple partition tables and file systems
# on devices.

# Default disk definitions for AWS
# --------------------------------
# (Not implemented yet, but provided for future documentation)

disk_setup:
  ephmeral0:
    table_type: 'mbr'
    layout: True
    overwrite: False

fs_setup:
  - label: None,
    filesystem: ext3
    device: ephemeral0
    partition: auto

# Default disk definitions for Microsoft Azure
# ------------------------------------------

device_aliases: {'ephemeral0': '/dev/sdb'}
disk_setup:
  ephemeral0:
    table_type: mbr
    layout: True
    overwrite: False

fs_setup:
  - label: ephemeral0
    filesystem: ext4
    device: ephemeral0.1
    replace_fs: ntfs


# Data disks definitions for Microsoft Azure
# ------------------------------------------

disk_setup:
  /dev/disk/azure/scsi1/lun0:
    table_type: gpt
    layout: True
    overwrite: True

fs_setup:
  - device: /dev/disk/azure/scsi1/lun0
    partition: 1
    filesystem: ext4


# Default disk definitions for SmartOS
# ------------------------------------

device_aliases: {'ephemeral0': '/dev/vdb'}
disk_setup:
  ephemeral0:
```

```
    table_type: mbr
    layout: False
    overwrite: False

fs_setup:
  - label: ephemeral0
    filesystem: ext4
    device: ephemeral0.0

# Caveat for SmartOS: if ephemeral disk is not defined, then the disk will
#    not be automatically added to the mounts.


# The default definition is used to make sure that the ephemeral storage is
# setup properly.

# "disk_setup": disk partitioning
# --------------------------------

# The disk_setup directive instructs Cloud-init to partition a disk. The format is:

disk_setup:
  ephmeral0:
    table_type: 'mbr'
    layout: 'auto'
  /dev/xvdh:
    table_type: 'mbr'
    layout:
      - 33
      - [33, 82]
      - 33
    overwrite: True

# The format is a list of dicts of dicts. The first value is the name of the
# device and the subsequent values define how to create and layout the
# partition.
# The general format is:
#   disk_setup:
#     <DEVICE>:
#       table_type: 'mbr'
#       layout: <LAYOUT|BOOL>
#       overwrite: <BOOL>
#
# Where:
#   <DEVICE>: The name of the device. 'ephemeralX' and 'swap' are special
#               values which are specific to the cloud. For these devices
#               Cloud-init will look up what the real devices is and then
#               use it.
#
#               For other devices, the kernel device name is used. At this
#               time only simply kernel devices are supported, meaning
#               that device mapper and other targets may not work.
#
#               Note: At this time, there is no handling or setup of
#               device mapper targets.
#
#   table_type=<TYPE>: Currently the following are supported:
#                   'mbr': default and setups a MS-DOS partition table
#                   'gpt': setups a GPT partition table
#
#               Note: At this time only 'mbr' and 'gpt' partition tables
#                   are allowed. It is anticipated in the future that
#                   we'll also have "RAID" to create a mdadm RAID.
#
#   layout={...}: The device layout. This is a list of values, with the
#               percentage of disk that partition will take.
#               Valid options are:
```

```
#                   [<SIZE>, [<SIZE>, <PART_TYPE]]
#
#               Where <SIZE> is the _percentage_ of the disk to use, while
#               <PART_TYPE> is the numerical value of the partition type.
#
#               The following setups two partitions, with the first
#               partition having a swap label, taking 1/3 of the disk space
#               and the remainder being used as the second partition.
#                 /dev/xvdh':
#                   table_type: 'mbr'
#                   layout:
#                     - [33,82]
#                     - 66
#                   overwrite: True
#
#               When layout is "true" it means single partition the entire
#               device.
#
#               When layout is "false" it means don't partition or ignore
#               existing partitioning.
#
#               If layout is set to "true" and overwrite is set to "false",
#               it will skip partitioning the device without a failure.
#
#   overwrite=<BOOL>: This describes whether to ride with saftey's on and
#               everything holstered.
#
#               'false' is the default, which means that:
#                   1. The device will be checked for a partition table
#                   2. The device will be checked for a file system
#                   3. If either a partition of file system is found, then
#                       the operation will be _skipped_.
#
#               'true' is cowboy mode. There are no checks and things are
#                   done blindly. USE with caution, you can do things you
#                   really, really don't want to do.
#
#
# fs_setup: Setup the file system
# ------------------------------
#
# fs_setup describes the how the file systems are supposed to look.

fs_setup:
  - label: ephemeral0
    filesystem: 'ext3'
    device: 'ephemeral0'
    partition: 'auto'
  - label: mylabl2
    filesystem: 'ext4'
    device: '/dev/xvda1'
  - cmd: mkfs -t %(filesystem)s -L %(label)s %(device)s
    label: mylabl3
    filesystem: 'btrfs'
    device: '/dev/xvdh'

# The general format is:
#   fs_setup:
#     - label: <LABEL>
#       filesystem: <FS_TYPE>
#       device: <DEVICE>
#       partition: <PART_VALUE>
#       overwrite: <OVERWRITE>
#       replace_fs: <FS_TYPE>
#
# Where:
#   <LABEL>: The file system label to be used. If set to None, no label is
```

```
#     used.
#
#   <FS_TYPE>: The file system type. It is assumed that the there
#     will be a "mkfs.<FS_TYPE>" that behaves likes "mkfs". On a standard
#     Ubuntu Cloud Image, this means that you have the option of ext{2,3,4},
#     and vfat by default.
#
#   <DEVICE>: The device name. Special names of 'ephemeralX' or 'swap'
#     are allowed and the actual device is acquired from the cloud datasource.
#     When using 'ephemeralX' (i.e. ephemeral0), make sure to leave the
#     label as 'ephemeralX' otherwise there may be issues with the mounting
#     of the ephemeral storage layer.
#
#     If you define the device as 'ephemeralX.Y' then Y will be interpetted
#     as a partition value. However, ephermalX.0 is the _same_ as ephemeralX.
#
#   <PART_VALUE>:
#     Partition definitions are overwriten if you use the '<DEVICE>.Y' notation.
#
#     The valid options are:
#     "auto|any": tell cloud-init not to care whether there is a partition
#       or not. Auto will use the first partition that does not contain a
#       file system already. In the absence of a partition table, it will
#       put it directly on the disk.
#
#       "auto": If a file system that matches the specification in terms of
#       label, type and device, then cloud-init will skip the creation of
#       the file system.
#
#       "any": If a file system that matches the file system type and device,
#       then cloud-init will skip the creation of the file system.
#
#       Devices are selected based on first-detected, starting with partitions
#       and then the raw disk. Consider the following:
#           NAME     FSTYPE LABEL
#           xvdb
#           |-xvdb1  ext4
#           |-xvdb2
#           |-xvdb3  btrfs  test
#           \-xvdb4  ext4   test
#
#         If you ask for 'auto', label of 'test, and file system of 'ext4'
#         then cloud-init will select the 2nd partition, even though there
#         is a partition match at the 4th partition.
#
#         If you ask for 'any' and a label of 'test', then cloud-init will
#         select the 1st partition.
#
#         If you ask for 'auto' and don't define label, then cloud-init will
#         select the 1st partition.
#
#         In general, if you have a specific partition configuration in mind,
#         you should define either the device or the partition number. 'auto'
#         and 'any' are specifically intended for formating ephemeral storage or
#         for simple schemes.
#
#       "none": Put the file system directly on the device.
#
#       <NUM>: where NUM is the actual partition number.
#
#   <OVERWRITE>: Defines whether or not to overwrite any existing
#     filesystem.
#
#     "true": Indiscriminately destroy any pre-existing file system. Use at
#         your own peril.
#
#     "false": If an existing file system exists, skip the creation.
```

```
#
#   <REPLACE_FS>: This is a special directive, used for Microsoft Azure that
#     instructs cloud-init to replace a file system of <FS_TYPE>. NOTE:
#     unless you define a label, this requires the use of the 'any' partition
#     directive.
#
# Behavior Caveat: The default behavior is to _check_ if the file system exists.
#   If a file system matches the specification, then the operation is a no-op.
```

Register Red Hat Subscription

```
#cloud-config

# register your Red Hat Enterprise Linux based operating system
#
# this cloud-init plugin is capable of registering by username
# and password *or* activation and org.  Following a successfully
# registration you can:
#   - auto-attach subscriptions
#   - set the service level
#   - add subscriptions based on its pool ID
#   - enable yum repositories based on its repo id
#   - disable yum repositories based on its repo id
#   - alter the rhsm_baseurl and server-hostname in the
#     /etc/rhsm/rhs.conf file

rh_subscription:
  username: joe@foo.bar

  ## Quote your password if it has symbols to be safe
  password: '1234abcd'

  ## If you prefer, you can use the activation key and
  ## org instead of username and password. Be sure to
  ## comment out username and password

  #activation-key: foobar
  #org: 12345

  ## Uncomment to auto-attach subscriptions to your system
  #auto-attach: True

  ## Uncomment to set the service level for your
  ##   subscriptions
  #service-level: self-support

  ## Uncomment to add pools (needs to be a list of IDs)
  #add-pool: []

  ## Uncomment to add or remove yum repos
  ##   (needs to be a list of repo IDs)
  #enable-repo: []
  #disable-repo: []

  ## Uncomment to alter the baseurl in /etc/rhsm/rhsm.conf
  #rhsm-baseurl: http://url

  ## Uncomment to alter the server hostname in
  ##  /etc/rhsm/rhsm.conf
  #server-hostname: foo.bar.com
```

Configure data sources

```
#cloud-config

# Documentation on data sources configuration options
datasource:
  # Ec2
  Ec2:
    # timeout: the timeout value for a request at metadata service
    timeout : 50
    # The length in seconds to wait before giving up on the metadata
    # service.  The actual total wait could be up to
    #   len(resolvable_metadata_urls)*timeout
    max_wait : 120

    #metadata_url: a list of URLs to check for metadata services
    metadata_urls:
     - http://169.254.169.254:80
     - http://instance-data:8773

  MAAS:
    timeout : 50
    max_wait : 120

    # there are no default values for metadata_url or oauth credentials
    # If no credentials are present, non-authed attempts will be made.
    metadata_url: http://mass-host.localdomain/source
    consumer_key: Xh234sdkljf
    token_key: kjfhgb3n
    token_secret: 24uysdfx1w4

  NoCloud:
    # default seedfrom is None
    # if found, then it should contain a url with:
    #    <url>/user-data and <url>/meta-data
    # seedfrom: http://my.example.com/i-abcde
    seedfrom: None

    # fs_label: the label on filesystems to be searched for NoCloud source
    fs_label: cidata

    # these are optional, but allow you to basically provide a datasource
    # right here
    user-data: |
      # This is the user-data verbatim
    meta-data:
      instance-id: i-87018aed
      local-hostname: myhost.internal

  Azure:
    agent_command: [service, walinuxagent, start]
    set_hostname: True
    hostname_bounce:
      interface: eth0
      policy: on # [can be 'on', 'off' or 'force']

  SmartOS:
    # For KVM guests:
    # Smart OS datasource works over a serial console interacting with
    # a server on the other end. By default, the second serial console is the
    # device. SmartOS also uses a serial timeout of 60 seconds.
    serial_device: /dev/ttyS1
    serial_timeout: 60

    # For LX-Brand Zones guests:
```

```
    # Smart OS datasource works over a socket interacting with
    # the host on the other end. By default, the socket file is in
    # the native .zoncontrol directory.
    metadata_sockfile: /native/.zonecontrol/metadata.sock

    # a list of keys that will not be base64 decoded even if base64_all
    no_base64_decode: ['root_authorized_keys', 'motd_sys_info',
                       'iptables_disable']
    # a plaintext, comma delimited list of keys whose values are b64 encoded
    base64_keys: []
    # a boolean indicating that all keys not in 'no_base64_decode' are encoded
    base64_all: False
```

Create partitions and filesystems

```
#cloud-config
# Cloud-init supports the creation of simple partition tables and file systems
# on devices.

# Default disk definitions for AWS
# --------------------------------
# (Not implemented yet, but provided for future documentation)

disk_setup:
  ephmeral0:
    table_type: 'mbr'
    layout: True
    overwrite: False

fs_setup:
  - label: None,
    filesystem: ext3
    device: ephemeral0
    partition: auto

# Default disk definitions for Microsoft Azure
# ------------------------------------------

device_aliases: {'ephemeral0': '/dev/sdb'}
disk_setup:
  ephemeral0:
    table_type: mbr
    layout: True
    overwrite: False

fs_setup:
  - label: ephemeral0
    filesystem: ext4
    device: ephemeral0.1
    replace_fs: ntfs


# Data disks definitions for Microsoft Azure
# ------------------------------------------

disk_setup:
  /dev/disk/azure/scsi1/lun0:
    table_type: gpt
    layout: True
    overwrite: True

fs_setup:
  - device: /dev/disk/azure/scsi1/lun0
```

```
    partition: 1
    filesystem: ext4


# Default disk definitions for SmartOS
# ------------------------------------

device_aliases: {'ephemeral0': '/dev/vdb'}
disk_setup:
  ephemeral0:
    table_type: mbr
    layout: False
    overwrite: False

fs_setup:
  - label: ephemeral0
    filesystem: ext4
    device: ephemeral0.0

# Caveat for SmartOS: if ephemeral disk is not defined, then the disk will
#    not be automatically added to the mounts.


# The default definition is used to make sure that the ephemeral storage is
# setup properly.

# "disk_setup": disk partitioning
# --------------------------------

# The disk_setup directive instructs Cloud-init to partition a disk. The format is:

disk_setup:
  ephmeral0:
    table_type: 'mbr'
    layout: 'auto'
  /dev/xvdh:
    table_type: 'mbr'
    layout:
      - 33
      - [33, 82]
      - 33
    overwrite: True

# The format is a list of dicts of dicts. The first value is the name of the
# device and the subsequent values define how to create and layout the
# partition.
# The general format is:
#   disk_setup:
#     <DEVICE>:
#       table_type: 'mbr'
#       layout: <LAYOUT|BOOL>
#       overwrite: <BOOL>
#
# Where:
#   <DEVICE>: The name of the device. 'ephemeralX' and 'swap' are special
#               values which are specific to the cloud. For these devices
#               Cloud-init will look up what the real devices is and then
#               use it.
#
#               For other devices, the kernel device name is used. At this
#               time only simply kernel devices are supported, meaning
#               that device mapper and other targets may not work.
#
#               Note: At this time, there is no handling or setup of
#               device mapper targets.
#
#   table_type=<TYPE>: Currently the following are supported:
```

```
#                   'mbr': default and setups a MS-DOS partition table
#                   'gpt': setups a GPT partition table
#
#               Note: At this time only 'mbr' and 'gpt' partition tables
#                   are allowed. It is anticipated in the future that
#                   we'll also have "RAID" to create a mdadm RAID.
#
#   layout={...}: The device layout. This is a list of values, with the
#               percentage of disk that partition will take.
#               Valid options are:
#                   [<SIZE>, [<SIZE>, <PART_TYPE]]
#
#               Where <SIZE> is the _percentage_ of the disk to use, while
#               <PART_TYPE> is the numerical value of the partition type.
#
#               The following setups two partitions, with the first
#               partition having a swap label, taking 1/3 of the disk space
#               and the remainder being used as the second partition.
#                 /dev/xvdh':
#                   table_type: 'mbr'
#                   layout:
#                     - [33,82]
#                     - 66
#                   overwrite: True
#
#               When layout is "true" it means single partition the entire
#               device.
#
#               When layout is "false" it means don't partition or ignore
#               existing partitioning.
#
#               If layout is set to "true" and overwrite is set to "false",
#               it will skip partitioning the device without a failure.
#
#   overwrite=<BOOL>: This describes whether to ride with saftey's on and
#               everything holstered.
#
#               'false' is the default, which means that:
#                   1. The device will be checked for a partition table
#                   2. The device will be checked for a file system
#                   3. If either a partition of file system is found, then
#                       the operation will be _skipped_.
#
#               'true' is cowboy mode. There are no checks and things are
#                   done blindly. USE with caution, you can do things you
#                   really, really don't want to do.
#
#
# fs_setup: Setup the file system
# ----------------------------
#
# fs_setup describes the how the file systems are supposed to look.

fs_setup:
  - label: ephemeral0
    filesystem: 'ext3'
    device: 'ephemeral0'
    partition: 'auto'
  - label: mylabl2
    filesystem: 'ext4'
    device: '/dev/xvda1'
  - cmd: mkfs -t %(filesystem)s -L %(label)s %(device)s
    label: mylabl3
    filesystem: 'btrfs'
    device: '/dev/xvdh'

# The general format is:
```

```
#   fs_setup:
#     - label: <LABEL>
#       filesystem: <FS_TYPE>
#       device: <DEVICE>
#       partition: <PART_VALUE>
#       overwrite: <OVERWRITE>
#       replace_fs: <FS_TYPE>
#
# Where:
#   <LABEL>: The file system label to be used. If set to None, no label is
#     used.
#
#   <FS_TYPE>: The file system type. It is assumed that the there
#     will be a "mkfs.<FS_TYPE>" that behaves likes "mkfs". On a standard
#     Ubuntu Cloud Image, this means that you have the option of ext{2,3,4},
#     and vfat by default.
#
#   <DEVICE>: The device name. Special names of 'ephemeralX' or 'swap'
#     are allowed and the actual device is acquired from the cloud datasource.
#     When using 'ephemeralX' (i.e. ephemeral0), make sure to leave the
#     label as 'ephemeralX' otherwise there may be issues with the mounting
#     of the ephemeral storage layer.
#
#     If you define the device as 'ephemeralX.Y' then Y will be interpetted
#     as a partition value. However, ephermalX.0 is the _same_ as ephemeralX.
#
#   <PART_VALUE>:
#     Partition definitions are overwriten if you use the '<DEVICE>.Y' notation.
#
#     The valid options are:
#     "auto|any": tell cloud-init not to care whether there is a partition
#       or not. Auto will use the first partition that does not contain a
#       file system already. In the absence of a partition table, it will
#       put it directly on the disk.
#
#       "auto": If a file system that matches the specification in terms of
#       label, type and device, then cloud-init will skip the creation of
#       the file system.
#
#       "any": If a file system that matches the file system type and device,
#       then cloud-init will skip the creation of the file system.
#
#       Devices are selected based on first-detected, starting with partitions
#       and then the raw disk. Consider the following:
#           NAME     FSTYPE LABEL
#           xvdb
#           |-xvdb1  ext4
#           |-xvdb2
#           |-xvdb3  btrfs  test
#           \-xvdb4  ext4   test
#
#         If you ask for 'auto', label of 'test, and file system of 'ext4'
#         then cloud-init will select the 2nd partition, even though there
#         is a partition match at the 4th partition.
#
#         If you ask for 'any' and a label of 'test', then cloud-init will
#         select the 1st partition.
#
#         If you ask for 'auto' and don't define label, then cloud-init will
#         select the 1st partition.
#
#         In general, if you have a specific partition configuration in mind,
#         you should define either the device or the partition number. 'auto'
#         and 'any' are specifically intended for formating ephemeral storage or
#         for simple schemes.
#
#       "none": Put the file system directly on the device.
```

```
#
#       <NUM>: where NUM is the actual partition number.
#
#   <OVERWRITE>: Defines whether or not to overwrite any existing
#     filesystem.
#
#     "true": Indiscriminately destroy any pre-existing file system. Use at
#         your own peril.
#
#     "false": If an existing file system exists, skip the creation.
#
#   <REPLACE_FS>: This is a special directive, used for Microsoft Azure that
#     instructs cloud-init to replace a file system of <FS_TYPE>. NOTE:
#     unless you define a label, this requires the use of the 'any' partition
#     directive.
#
# Behavior Caveat: The default behavior is to _check_ if the file system exists.
#   If a file system matches the specification, then the operation is a no-op.
```

Grow partitions

```
#cloud-config
#
# growpart entry is a dict, if it is not present at all
# in config, then the default is used ({'mode': 'auto', 'devices': ['/']})
#
#  mode:
#    values:
#     * auto: use any option possible (any available)
#             if none are available, do not warn, but debug.
#     * growpart: use growpart to grow partitions
#             if growpart is not available, this is an error.
#     * off, false
#
# devices:
#   a list of things to resize.
#   items can be filesystem paths or devices (in /dev)
#   examples:
#     devices: [/, /dev/vdb1]
#
# ignore_growroot_disabled:
#   a boolean, default is false.
#   if the file /etc/growroot-disabled exists, then cloud-init will not grow
#   the root partition.  This is to allow a single file to disable both
#   cloud-initramfs-growroot and cloud-init's growroot support.
#
#   true indicates that /etc/growroot-disabled should be ignored
#
growpart:
  mode: auto
  devices: ['/']
  ignore_growroot_disabled: false
```