c0mbat - Product Documentation

Release 3.7a

Weqaar Janjua

Jan 09, 2019

CHAPTER

ONE

INTRODUCTION

1.1 General Introduction

c0mbat is a Configuration Management tool developed with Slack.

c0mbat is short for Zero-Configuration Management and Build Automorphic Tool.

1.2 Product

c0mbat is intended for configuration management in Cloud Infrastructures.

- Features
  1. Multi-threaded (uses multiple cores as scheduled by the OS or by user i.e. numa utils)
  2. Parallel deployment currently over SSH channel
  3. Application wide configuration file: c0mbat.conf
  4. APT package manager handling: install, uninstall (remove), update
  5. Services handling: start, stop, restart, reload
  6. Inventory and Artifacts handled seprately under dir "deploy"
     (a) deploy/artifacts
     (b) deploy/inventory
  7. Idempotency with deployments: keeps MD5 Hashes as deployment cache for each host and it's artifact relationships
     (a) Filesystem hash of individual artifacts under deploy/artifacts
     (b) Configuration hash of individual artifacts, defined in deploy/artifacts/artifacts.json
     (c) Hashes are maintained as cache under "cache/cache.json" that is built after initial run
     (d) In ideal situation this should be a distributed cache accessible over IP i.e. NoSQL DB, to allow multiple client's to sync
  8. Two types of deployments currently supported
     (a) APT package manager based
     (b) Manual
  9. Manual type deployments support Permssions, this is specified optionally as artifact's metadata (see deploy/artifacts/artifacts.json)

10. Logging under dir “logs/”
11. Cleanup runtime/transient data with the script “clean.sh”. Note this cleans up the logs and cache as well.

1.3 Availability of code

Available on Github at c0mbat repo

1.4 Architectrure / Code Walk-through

1.4.1 Structure

Code is organized in this hierarchy

Code is functionally divided as packages

```
src/packages
├── artifacts
│   ├── initartificats.py
│   └── __init__.py
├── conf
│   ├── configinit.py
│   ├── __init__.py
│   └── sysconfigx.py
├── inventory
│   ├── initinventory.py
│   └── __init__.py
├── queue
│   ├── __init__.py
│   └── initqueues.py
├── remoteaccess
│   ├── __init__.py
│   └── ssh.py
├── runtime
│   ├── cache.py
│   ├── __init__.py
│   └── sysinit.py
├── threads
│   ├── deploymentworker.py
│   └── __init__.py
├── globalvars.py
└── __init__.py
```

1.4.2 Code re-usability

Source code allows for easy reuse of various components

1. All variables and objects are initailized as run-time from "packages/conf/configinit.py"
2. Initialized objects are available at run-time in "packages/globalvars.py"
3. Re-use is as simple as calling the desired object with globalvars.<object>
   (a) Example: to log a debug message use - globalvars._stats_logger.debug("YOUR MESSAGE!")

1.5 Usage

1. Installation instructions README.txt
2. c0mbat provides a CLI Interface, run with: python c0mbat.py -h

1.5.1 Inventory

Hosts or Nodes are defined in the inventory file "inventory.json" available under "deploy/inventory/" directory.

Format is::

```
"Host" : {
    "Address" : "0.0.0.0",
    "Auth" : { "username": "root", "password" : "r00t" },
    "disabled" : false,
    "Artifacts" : ["artifact-1", "artifact-2"]
},
```

1.5.2 Artifacts

Artifacts are the source files and/or directories that need deployment on a remote node. These are defined in the artifacts file "artifacts.json" available under "deploy/artifacts/" directory.

Format is::

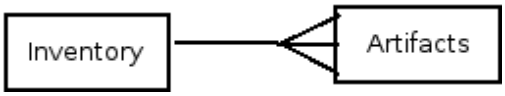
```
"Apache" : {
     "pkg-type": "apt",
     "pkg-name" : "apache2",
     "pkg-action" : "install",
     "pre-deploy-trigger" : "update",
     "post-deploy-trigger" : "restart",
     "service-name" : "apache2"
 },
 "php-app" : {
     "pkg-type": "manual",
     "pkg-action" : "install",
     "pkg-path" : "/var/www/html",
     "dependency" : {
         "service-name" : "apache2",
         "pre-deploy-trigger" : null,
         "post-deploy-trigger" : "restart"
     },
     "metadata" : {
     "owner" : "www-data",
     "group" : "www-data",
     "mode" : "644"
     }
 }
```

Artifacts of type "manual" must then be placed under filesystem path as under::

```
deploy/artifacts/php-app/
```

1.5.3 Relationship between 'Inventory' and 'Artifacts'

There is one-to-many relationship between Inventory and Artifact elements.



To clarify this relationship, a single node may relate to many Artifacts. As an example:

- Inventory:

```
"Zebra" : {
    "Address" : "10.10.10.10",
    "Auth" : { "username": "root", "password" : "r00t" },
    "disabled" : false,
    "Artifacts" : ["Apache", "php-app"]
},
```

- Artifacts:

```
"Apache" : {
     "pkg-type": "apt",
     "pkg-name" : "apache2",
     "pkg-action" : "install",
     "pre-deploy-trigger" : "update",
     "post-deploy-trigger" : "restart",
     "service-name" : "apache2"
 },
 "php-app" : {
     "pkg-type": "manual",
     "pkg-action" : "install",
     "pkg-path" : "/var/www/html",
     "dependency" : {
         "service-name" : "apache2",
         "pre-deploy-trigger" : null,
         "post-deploy-trigger" : "restart"
     },
     "metadata" : {
     "owner" : "www-data",
     "group" : "www-data",
     "mode" : "644"
     }
 }
```

Host “Zebra” needs to be deployed Artifacts “Apache” and “php-app”, thus it relates to both of them.

CHAPTER

TWO

SUPPORT

Note: Contact weqaar.janjua@gmail.com