gotun Documentation

Release 0.1

Kushal Das

Apr 03, 2017

Contents

This is written from scratch version of Tunir in golang.

Contents:

CHAPTER 1

Install Instructions

gotun is written in golang. To install the tool, you will need golang on your system.

Install Go & setup the environment

You can install from your distro's package, or you can install the upstream package. After that we will create a workspace *~/gocode/*. Add the following in your *~/.bashrc* file, and then source it.

```
export PATH=~/gocode/bin:$PATH
export GOPATH=~/gocode/
```

Install gotun

```
$ go get github.com/kushaldas/gotun
```

After this you should have the gotun binary in the *~/gocode/bin/* directory.

CHAPTER 2

Configuration of Jobs

gotun expects the job configuration in a yaml file. The following are two different examples of the job. Each job has two files, one is the yaml file which contains the configuration (say AWS or Openstack), and also the jobname.txt file which contains the commands to execute.

OpenStack based job

```
---
BACKEND: "openstack"

OS_AUTH_URL: "URL"
TENANT_ID: "Your tenant id"
USERNAME: "USERNAME"
PASSWORD: "PASSWORD"
OS_REGION_NAME: "RegionOne"
OS_IMAGE: "Fedora-Atomic-24-20161031.0.x86_64.qcow2"
OS_FLAVOR: "m1.medium"
OS_SECURITY_GROUPS:
    - "group1"
    - "default"
OS_NETWORK: "NETWORK_POOL_ID"
OS_FLOATING_POOL: "POOL_NAME"
OS_KEYPAIR: "KEYPAIR NAME"
key: "Full path to the private key (.pem file)"
```

In the above example *gotun* expects the Image is already available in the cloud. If you want to upload a new image for the test, and then delete it after the test, then provide a full path to the image .qcow2 file in *OS_IMAGE*.

```
OS_IMAGE: "/home/kdas/Fedora-Atomic-24-20161031.0.x86_64.qcow2"
```

You can also set the following environment variables for the OpenStack job.

- OS_TENANT_ID

- OS_USERNAME
- OS_PASSWORD

User data on OpenStack

You can provide path to a cloud-config userdata file in the configuration file. The following line expects a proper YAML file in the given location.

```
user-data: "/home/user/work/data.yml"
```

You can learn more about cloud-init (userdata) examples here.

Multiple VM(s) on OpenStack

In case you want to spin up more than one vm on OpenStack, then add a *NUMBER* value to the yml file.

```
NUMBER: 3
```

AWS based job

```
---
BACKEND: "aws"

AWS_AMI: "ami-df3367bf"
AWS_INSTANCE: "t2.medium"
AWS_KEYNAME: "The name of the key"
AWS_SUBNET:  "subnet-ID"
AWS_SECURITYGROUPIDS:
    - "sg-groupid"
AWS_REGION: "us-west-1"
USERKEY: "YOURKEY"
SECRET: "SECRET KEY PART"
key: "PATH to the .pem file"
```

Update the configuration based on your need. You can see that you will need to find subnet-id, security group ids for each region to work with.

You can also set the following environment variables for the AWS job.

- AWS_USERKEY
- AWS_SECRET

For AWS based jobs, one can also pass the *AMI_ID* and *REGION* by command line arguments. The following two new flags were added for the same.

--ami-id value the AMI ID for AWS jobs

--region value Region name for AWS based jobs

For remote systems

```
---
BACKEND: "bare"
key: "Path to the .pem file"
PORT: 22
USER: "username"
VMS:
    vm1: IP1
    vm2: IP2
```

The keys of *VMS* are the vm numbers, you will have to mark at least *vm1* and the corresponding IP address.

Note: The default username is *fedora*, and default port is *22*.

CHAPTER 3

Detailed description of the tests

The *jobname.txt* text file contains the bash commands to run in the system, one command per line. In case you are rebooting the system, you may want to use **SLEEP NUMBER_OF_SECONDS** directive there.

If a command starts with @@ sign, it means the command is supposed to fail. Generally, we check the return codes of the commands to find if it failed, or not. For Docker container-based systems, we track the stderr output.

We can also have non-gating tests, means these tests can pass or fail, but the whole job status will depend on other gating tests. Any command in jobname.txt starting with ## sign will mark the test as non-gating.

Example:

```
## curl -O https://kushal.fedorapeople.org/tunirtests.tar.gz
ls /
## foobar
## ls /root
##  sudo ls /root
date
@@ sudo reboot
SLEEP 40
ls /etc
```

Polling the VM(s)

We can use the *POLL* directive in the jobfile after a reboot, this will try to POLL every VM to make sure that we have the SSH service back in all the VM((s).

Example:

```
ls
@@ sudo reboot
POLL
ls /
```

Coping files over

gotun can copy files over to any VM using scp. The following is an example where we are copying a binary file into bin directory inside home of the user on vm1.

```
COPY: localfile.bin vm1:./bin/
```

Multiple VM based tests

In case of tests containing multiple VM(s), one mark the tests with vm numbers. This way, we decide which test will run on which vm. The numbers start from *vm1* to *vm9*.

Example:

```
vm1 wget https://kushaldas.in
vm2 sudo mkdir /root/hello_dir
vm1 sudo dnf install pss -y
vm1 which pss
```

If no vm number is marked at the begining of any line, gotun assumes that the test is supposed to run on *vm1*.

Rebuild of VM(s) on OpenStack

Note: This feature is only available for OpenStack based jobs. For other kind of tests, this will do nothing.

REBUILD_SERVERS directive will rebuild all of the avialable VM(s) on OpenStack. They will try to POLL the VM(s) after rebuilding them. This step is sequential for now. In future, we will be doing this in parallal.

```
echo "hello asd" > ./hello.txt
vm1 sudo cat /etc/machine-id
mkdir {push,pull}
ls -l ./
pwd
REBUILD_SERVERS
sudo cat /etc/machine-id
ls -l ./
pwd
```

The following is the output from the above mentioned test.

```
$ gotun --job fedora
Starts a new Tunir Job.

Server ID: e0d7b55a-f066-4ff8-923c-582f3c9be29b
Let us wait for the server to be in running state.
Time to assign a floating pointip.
Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.
```

```
Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Server ID: a0b810e6-0d7f-4c9e-bc4d-1e62b082673d
Let us wait for the server to be in running state.
Time to assign a floating pointip.
Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Executing:  echo "hello asd" > ./hello.txt
Executing:  vm1 sudo cat /etc/machine-id
Executing:  mkdir {push,pull}
Executing:  ls -l ./
Executing:  pwd
Going to rebuild: 209.132.184.241
Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Going to rebuild: 209.132.184.242
Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Polling for a successful ssh connection.

Executing:  sudo cat /etc/machine-id
Executing:  ls -l ./
Executing:  pwd

Result file at: /tmp/tunirresult_180507156
```

```
Job status: true


command: echo "hello asd" > ./hello.txt
status:true



command: sudo cat /etc/machine-id
status:true

e0d7b55af0664ff8923c582f3c9be29b


command: mkdir {push,pull}
status:true



command: ls -l ./
status:true

total 4
-rw-rw-r--. 1 fedora fedora 10 Jan 25 13:58 hello.txt
drwxrwxr-x. 2 fedora fedora  6 Jan 25 13:58 pull
drwxrwxr-x. 2 fedora fedora  6 Jan 25 13:58 push


command: pwd
status:true

/var/home/fedora


command: sudo cat /etc/machine-id
status:true

e0d7b55af0664ff8923c582f3c9be29b


command: ls -l ./
status:true

total 0


command: pwd
status:true

/var/home/fedora


Total Number of Tests:8
Total NonGating Tests:0
Total Failed Non Gating Tests:0
```

```
Success.
```

Creating inventory file for Ansible based tests

Ansible is a powerful choice with many different usecases. One such usecase is about testing. Sometimes we just setup the whole test environment using Ansible, and some other times the whole testsuite is written on top of ansible. To enable using of predefined Ansible playbooks, gotun provides a file *current_run_info.json* for each run of job. This file contains a dictionary of vm numbers, and corresponding IP address, and also the *keyfile* value with the path of the private keyfile. This can be used with a simple Python or shell script to create the actual inventory file. For example, the following script *createinventory.py* will create a file called *inventory* in the current directory, and it assumes that there will be 2 VM(s) are avaiable (means it is running on OpenStack).

```
#!/usr/bin/env python3
import json

data = None
with open("current_run_info.json") as fobj:
    data = json.loads(fobj.read())

user = data['user']
host1 = data['vm1']
host2 = data['vm2']
key = data['keyfile']

result = """{0} ansible_ssh_host={1} ansible_ssh_user={2} ansible_ssh_private_key_
↪file={3}
{4} ansible_ssh_host={5} ansible_ssh_user={6} ansible_ssh_private_key_file={7}""".
↪format(host1,host1,user,key,host2,host2,user,key)
with open("inventory", "w") as fobj:
    fobj.write(result)
```

As you can see, we are reading the *current_run_info.json* file first, and then creating a file called *inventory*. We can then execute this script by using the *HOSTCOMMAND* directive in the test.

```
HOSTCOMMAND: ./createinventory.py
```

Running Ansible on the HOST as part of a test

The next step is to run Ansible playbook on the host system as a test. This can be done with a *HOSTTEST* directive. The following example test file will first create the inventory file using a *HOSTCOMMAND* directive, and then execute the an ansible playbook.

```
HOSTCOMMAND: ./onevm.py
HOSTTEST: ansible-playbook -b -i inventory atomic-host-tests/tests/improved-sanity-
↪test/main.yml
```

CHAPTER 4

Examples of usage

Here are a few blog posts explaining various example use-cases of gotun.

- Testing redis containers on Fedora Atomic
- Testing MariaDB conatainer on Fedora
- Using Ansible inside of gotun

CHAPTER 5

Indices and tables

- genindex
- modindex
- search