# Interacting with Tagged EC2 Instances using Fabric

Abhishek Tiwari

Citation: *A. Tiwari*, "Interacting with Tagged EC2 Instances using Fabric", Abhishek Tiwari, 2014. doi:10.59350/rt9hs-mfv44

Published on: July 01, 2014

AWS allows tagging of EC2 resources using your own metadata (key-value pair). Tagging of EC2 resources is quite helpful in managing instances, images and organise billing by tag names. An EC2 instance can be tagged by CLI tools, API wrappers or using AWS Console when launching the instance as well as after launching the instance.

## Getting Started

In this post we are gone discuss how you can interact with already tagged EC2 instances using Fabric. Fabric is a Python package and command-line tool. Fabric simplifies and streamlines administration of large number of server. Fabric provides operations to execute local or remote shell commands in parallel.

First lets install `boto` and `Fabric` using `pip`. boto is Python client for AWS. ~~~ pip install boto pip install fabric ~~~

Alternatively you can install `boto` and `Fabric` using your OS package manager. Remember sometime OS package name can start with `python-`, so `python-boto` or `python-fabric`.

```
sudo apt-get install fabric
sudo apt-get install boto
```

I am assuming that you have downloaded AWS private key file (`.pem`) for the key pair that you specified when you launched the instance. After download you can move this `.pem` file to a secure location, ideally `.ssh/` folder and set appropriate permission. If you have more than one AWS private key files then follow same steps for all of them.

```
mv ~/Downloads/mykey.pem ~/.ssh/mykeypair_1.pem
chmod 400 ~/.ssh/mykeypair_1.pem
```

Next step is to add your AWS access key and secret access key in your local environment. Plus we are also setting default AWS region.

```
export AWS_EC2_REGION=us-west-1
export AWS_ACCESS_KEY_ID=<your-access-key>
export AWS_SECRET_ACCESS_KEY=<your-secret-key>
```

Now we can write our Fabric script `fabfile.py`. A `fabfile.py` is collection of Fabric tasks and private functions. Fabric tasks are standard Python functions and they invoke private functions provided by Fabric library such as `sudo`, `with`, `cd`, etc. Private function name starts with underscore (_). Fabric tasks can be listed on command line but not private functions. In a nutshell, our `fabfile.py` looks like something like following.

```
import boto, urllib2
from   boto.ec2 import connect_to_region
```

```
from   fabric.api import env, run, cd, settings, sudo
from   fabric.api import parallel
import os
import sys

REGION       = os.environ.get("AWS_EC2_REGION")
WEB_ROOT = "/var/www"

# Server user, normally AWS Ubuntu instances have default user "ubuntu"
env.user      = "ubuntu"

# List of AWS private key Files
env.key_filename = ["~/.ssh/mykeypair_1.pem", "~/.ssh/mykeypair_2.pem"]

# Fabric task to restart Apache, runs in parallel
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 restart_apache
@parallel
def reload_apache():
    sudo('service apache restart')

# Fabric task to start Apache, runs in parallel
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 start_apache
@parallel
def start_apache():
    sudo('service apache start')

# Fabric task to stop Apache, runs in parallel
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 stop_apache
@parallel
def stop_apache():
    sudo('service apache stop')

# Fabric task to updates/upgrade OS (Ubuntu), runs in parallel
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 update_os
@parallel
def update_os():
    sudo('apt-get update -y')
    sudo('apt-get upgrade -y')

# Fabric task to reboot OS (Ubuntu), runs in parallel
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 reboot_os
@parallel
def reboot_os():
    sudo('reboot')
```

```
# Fabric task for cloning GIT repository in Apache WEB_ROOT
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 update_branch restart_apache
@parallel
def clone_branch():
    with cd("/var/www"):
        run('git clone https://www.github.com/user/repo.git')

# Fabric task for deploying latest changes using GIT pull
# This assumes that your GIT repository is in Apache WEB_ROOT
# To execute task using fabric run following
# fab set_hosts:phpapp,2X,us-west-1 update_branch restart_apache
@parallel
def update_branch():
    with cd("/var/www"):
        run('git pull -f')

# Your custom Fabric task here after and run them using,
# fab set_hosts:phpapp,2X,us-west-1 task1 task2 task3

# Fabric task to set env.hosts based on tag key-value pair
def set_hosts(tag = "phpapp", value="*", region=REGION):
    key              = "tag:"+tag
    env.hosts    = _get_public_dns(region, key, value)

# Private method to get public DNS name for instance with given tag key
   and value pair
def _get_public_dns(region, key, value ="*"):
    public_dns   = []
    connection   = _create_connection(region)
    reservations = connection.get_all_instances(filters = {key : value})
    for reservation in reservations:
        for instance in reservation.instances:
            print "Instance", instance.public_dns_name
            public_dns.append(str(instance.public_dns_name))
    return public_dns

# Private method for getting AWS connection
def _create_connection(region):
    print "Connecting to ", region

    conn = connect_to_region(
        region_name = region,
        aws_access_key_id=os.environ.get("AWS_ACCESS_KEY_ID"),
        aws_secret_access_key=os.environ.get("AWS_SECRET_ACCESS_KEY")
    )

    print "Connection with AWS established"
    return connection
```

This Fabric script documentation is pretty self explanatory. To run tasks using above Fabric script you

will run `fab` command (using your favourite terminal in same directory as `fabfile.py`) with lists of tasks separated by space. Depending on task definition you can pass a list of comma separated parameters or named parameters after task name. Task name and parameters are separated by `:`. Named parameters are passed like `parameter=value`. Let see some examples of how we can execute Fabric tasks.

Following command will execute Fabric tasks in sequence `task1` → `task2` → `task3`.

```
fab task1 task2 task3
```

Next command will execute Fabric tasks in sequence `set_hosts` → `task1` → `task2` → `task3`. For `set_hosts` task we are passing 3 parameter `phpapp`, `2X`, `us-west-1` to `set_hosts` task those maps on function parameter `tag`, `value` and `region`.

```
fab set_hosts:phpapp,2X,us-west-1 task1 task2 task3
```

A named parameter version of above example will look like,

```
fab set_hosts:tag=phpapp,value=2X,region=us-west-1 task1 task2 task3
```

In next example we are only passing only `tag`, other two parameter default to `*` and `us-west-1` according to function definition.

```
fab set_hosts:phpapp task1 task2 task3
```

### tl;dr

During `set_hosts` task, Fabric is using AWS credentials from OS environment and region name to create an API connection. Then trying to get host names for AWS instances with matching key-value pair metadata using this established API connection. This is accomplished by first getting all instances and then filtering them on tag key-value pair.

Once `set_hosts` task is complete, Fabric will have a list of host names to run remaining task in sequence `task1` → `task2` → `task3` on each instance. Fabric will SSH into each AWS instance using private keys listed in `env.key_filename` of this Fabric file. Please note that `@parallel` decorator allows to run tasks in parallel on multiple instances. Use `@parallel` when appropriate, `@parallel` don't change sequence of task run, it just facilitating you to run commands in non-blocking manner on multiple instances.

How about some real world example? Here you go, you can run following command to update the GIT repo on all instances in Apache root using GIT pull and restart the Apache after update.

```
fab set_hosts:phpapp,2X,us-west-1 update_branch restart_apache
```

That's it. You can always add your own custom task on top of this Fabric wrapper.