Base Environment Generator

This set of scripts is intended to provide a common and consistent environment design for building out complex demo environments and has utility both as a Python class as well as a command line tool.

This python script will create a VPC that should deploy cleanly into any region across the publicly available set of AWS regions worldwide. This network, by default, will include:

* A public (/24) and a private subnet (/22) in two different Availability Zones
* A NAT instance per AZ
* A bastion host
* An S3 bucket configured to allow Amazon ELB (within the same region) and AWS CloudTrail to aggregate logs

There are a number of configuration options documented within the script itself using [docopt](http://docopt.org). An overview of the general capabilities and features is as follows:

* This script queries the AWS VPC API to ensure that the AZ's selected for deployment will allow subnets to be deployed to them (sometimes an issue in older accounts)
* Modify the base network address, size and subnet size and count via parameters
* Set key name prefixes for ELB and CloudTrail logging paths within the created bucket

Used from the command line, this will generate the network alone, but used as a base class, it's a powerful tool to help generate the basic structures for more complex demo environments in CloudFormation very easily.

## Python Usage

This class is designed to be useful for command line usage, but has further utility as a base class for more complicated environments. The original aim of this script was to build a reusable artifact that could serve as the common networking design for multi-AZ, multi-subnet demo environments. As such, the environmentbase.py script contains a number of methods that are meant to be used by parent classes and provide abstractions for common workflows and use cases. To use this class, simply place it in your project folder (this is not yet packaged in a deployable artifact). The import and class definition for your class will look similar to the following:

from EnvBase.environmentbase import EnvironmentBase

class ElkDemo(EnvironmentBase):

def \_\_init\_\_(self, \*\*kwargs):

EnvironmentBase.\_\_init\_\_(self, \*\*kwargs)

Documentation within the class takes a modified usage of the [doxygen](http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html#pythonblocks) standard by adding a @kwarg identifier that indicates that a given method utilizes an argument that's passed in via the class constructor along with the type and description of that parameter.

## Command Line Usage

To use this script, you must install the following Python libraries:

* [docopt](http://docopt.org)
* [boto](http://boto.readthedocs.org/en/latest/)
* [troposphere](https://github.com/cloudtools/troposphere)
* [ipcalc](https://pypi.python.org/pypi/ipcalc/)

This can be done by running the following command from this directory:

sudo pip install -r requirements.txt –upgrade

To use the script itself, you can run it directly from the command line:

python environmentbase.py –h

This script uses [docopt](http://docopt.org) to collect and parse arguments per the help documentation included. You do have the option of either passing in a configuration file representing these values or using the command line arguments to set them individually. A sample (with defaults) of the configuration file is located in the config\_args.json file. To run this, you must have a boto.cfg file set with the credentials for the target account where you'd like to deploy the CloudFormation template to.

The IAM permissions required to perform the VPC lookups are the following:

{"Statement": [

{"Action": ["ec2:DescribeAvailabilityZones", "ec2:DescribeRegions"],

"Effect": "Allow",

"Resource": "\*"

}]}

Once you have either set up your boto.cfg file or have gathered an access key and secret key for a user with the requisite permissions, you can run the process as follows:

python environmentbase.py create --config config\_args.json

**or**

python environmentbase.py create --aws\_access\_key\_id <ACCESS\_KEY\_ID> --aws\_secret\_access\_key <SECRET\_ACCESS\_KEY>

## File Descriptions

#### ****init****.py

This file is in place to help references from projects that consume the environmentbase.py class. This allows for this entire folder to be placed within an existing python project so that it can be properly referenced by the consuming class. References to this class from other classes where this folder of files is present should be done as follows (where the name of the folder where this file package sits is **/EnvBase**:

from EnvBase.environmentbase import EnvironmentBase

#### ami\_cache.json

This file is a simple JSON dictionary to use when assembling the 'RegionMap' object within the CloudFormation template. This is a simple way to abstract AMI Id's out of the solution and can be used in conjunction with Packer to populate custom AMIs directly from external tools.

#### config\_args.json

This is a sample 'config' file to be passed into the command line use case and identifies a simple dictionary of configuration keys and values to be used within the execution of the script.

#### environmentbase.py

This is the actual python script and can be used as a stand-alone class or as a command line tool. Documentation is in pseudo-[doxygen](http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html#pythonblocks) format and the command line argument parsing process is implemented in [docopt](http://docopt.org). This script requires that all packages in the requirements.txt file are installed.

#### environmentbase.template

This is a sample output from the environmentbase.py script run from the command line with all arguments set to their defaults.

#### EnvironmentBase.docx

This documentation file.

#### README.md

Markdown representation of file--documentation for usage of this set of scripts and files.

#### requirements.txt

Python pip-formatted requirements file to be used to install the prerequisites to run the environmentbase.py script either as a base class within another project or as a command line tool. To install the required packages, run the following command in the command line referencing this file:

sudo pip install -r requirements.txt --upgrade