

---

# ZK Watcher Documentation

*Release*

**Nextdoor Engineering**

December 17, 2015



|          |                           |          |
|----------|---------------------------|----------|
| <b>1</b> | <b>About</b>              | <b>1</b> |
| 1.1      | Installation . . . . .    | 1        |
| 1.2      | Running . . . . .         | 2        |
| 1.3      | Authentication . . . . .  | 4        |
| <b>2</b> | <b>Indices and tables</b> | <b>5</b> |



---

## About

---

`zk_watcher` is a simple service that registers [Ephemeral Nodes](#) in Apache Zookeeper based on the result of a healthcheck. The service is available both as a Python script that you can run on your own, or as a Docker image ([nextdoor/zkwatcher](#)) that you can pull down.

The goal of `zk_watcher` is to monitor a particular service and register that machine as a `provider` of that service at a given path on the Zookeeper service.

A simple example is having `zk_watcher` monitor Apache `httpd` by running `service apache2 status` at a regular interval and registers with ZooKeeper at a given path (say `/services/production/webserver`). As long as the command returns a safe exit code (0), `zk_watcher` will register with ZooKeeper that this server is providing this particular service. If the hostname of the machine is `web1.mydomain.com`, the registration path would look like this

```
/services/production/webserver/web1.mydomain.com:80
```

In the event that the service check fails, the host will be immediately de-registered from that path.

## 1.1 Installation

### 1.1.1 Local Python Install

To install the application locally, run :

```
$ python setup.py install
```

or

```
$ pip install zk_watcher
```

### 1.1.2 Docker Container Installation

You can pull down the latest builds of `zk_watcher` built into a fully self-sufficient Docker image like this:

```
$ docker pull nextdoor/zkwatcher
Using default tag: latest
latest: Pulling from nextdoor/zkwatcher
Digest: sha256:47eee56494a190e35c5d25d2285056d0e1e347ee276d7792973fb803511da00a
Status: Image is up to date for nextdoor/zkwatcher:latest
```

## 1.2 Running

### 1.2.1 Docker Execution

When running as a Docker container, you can use `nextdoor/zkwatcher` to monitor a *single* service. The service is configured by passing in the following `ENVIRONMENT` variables into your Docker runtime.

#### Required Variables

- `ZK_PATH`: The path in Zookeeper to register the Ephemeral node.
- `SVC_HOST`: The hostname/IP that will be registered under `ZK_PATH`
- `SVC_PORT`: The port that will be registered along with the `SVC_HOST`
- `CMD`: The command that will be called to check the service.

#### Optional Variables

- `REFRESH`: The time to wait between executions of the `CMD` (default: 30)
- `ZOOKEEPER_HOST`: The Zookeeper Host/IP Endpoint (default: `$DOCKER_HOST_IP`)
- `ZOOKEEPER_PORT`: The Zookeeper TCP Port (default: 2181)
- `VERBOSE`: Set to `true` to enable verbose logging.

#### Dynamically Populated Variables

- `DOCKER_HOST_IP`: This variable is dynamically generated and is the **docker host** IP address that the container sees. It effectively works out to `localhost` inside a normal host OS.

#### Checking a Docker Host Service

The one interesting bit about this execution is the `CMD` below. We pass in an `_escaped_` variable, and that variable will be evaluated later once the container actually starts. In this case, it allows us to let the container use one of the above dynamically populated variables.

```
$ docker run \  
--env ZK_PATH=/ssh_services \  
--env SVC_HOST=$(hostname -f) \  
--env SVC_PORT=22 \  
--env CMD="nc -v -z -w 1 \${DOCKER_HOST_IP} 22" \  
zk_watcher  
  
Starting zk_watcher up with the following config:  
[service]  
cmd: nc -v -z -w 1 172.17.0.1 22  
refresh: 30  
service_port: 22  
service_hostname: vagrant-ubuntu-trusty-64  
zookeeper_path: /ssh_services  
zk_watcher[16] [WatcherDaemon] [__init__]: (INFO) WatcherDaemon 0.3.2  
zk_watcher[16] [nd_service_registry] [__init__]: (INFO) Initializing ServiceRegistry object  
zk_watcher[16] [nd_service_registry] [_connect]: (INFO) Connecting to Zookeeper Service (172.17.0.1:22)  
zk_watcher[16] [nd_service_registry] [_state_listener]: (INFO) Zookeeper connection state changed: Connected  
zk_watcher[16] [nd_service_registry] [__init__]: (INFO) Initialization Done!  
172.17.0.1 (172.17.0.1:22) open  
zk_watcher[16] [nd_service_registry.registration] [_create_node]: (INFO) [/ssh_services/vagrant-ubuntu-trusty-64]
```

#### Monitoring a different Docker Container

The more likely use of this container is to monitor a separate container and register that in Zookeeper. Here's a simple example of registering an Apache "hello world" container. We make use of Docker container linking and the variables that they create for you (\$APACHE\_PORT\_80\_TCP\_ADDR and \$APACHE\_PORT\_80\_TCP\_PORT in this case) to discover the hello-world containers IP and Port.

```
$ docker run -d --name hello-world tutum/hello-world
77a83d2be90f52541b1c8e54e5895a0d0c435d07af2da87d288693f54976e232
vagrant@vagrant-ubuntu-trusty-64:~/src$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS             
77a83d2be90f        tutum/hello-world  "/bin/sh -c 'php-fpm "  2 seconds ago      Up 1 seconds
$ docker run \
--env SVC_HOST=$(hostname -f) \
--env SVC_PORT=80 \
--env CMD="curl --fail http://\${APACHE_PORT_80_TCP_ADDR}:\${APACHE_PORT_80_TCP_PORT}" \
--env ZK_PATH=/hello_world --link "hello-world:apache"
zk_watcher

Starting zk_watcher up with the following config:
[service]
cmd: curl --fail http://172.17.0.2:80
refresh: 30
service_port: 80
service_hostname: vagrant-ubuntu-trusty-64
zookeeper_path: /hello_world
zk_watcher[16] [WatcherDaemon] [__init__]: (INFO) WatcherDaemon 0.3.2
zk_watcher[16] [nd_service_registry] [__init__]: (INFO) Initializing ServiceRegistry object
zk_watcher[16] [nd_service_registry] [_connect]: (INFO) Connecting to Zookeeper Service (172.17.0.1:2181)
zk_watcher[16] [nd_service_registry] [_state_listener]: (INFO) Zookeeper connection state changed: Connected
zk_watcher[16] [nd_service_registry] [__init__]: (INFO) Initialization Done!
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
 Dload  Upload  Total    Spent    Left  Speed
100    478    0    478    0    0  53871    0  ---:---:---  ---:---:---  ---:---:--- 59750
zk_watcher[16] [nd_service_registry.registration] [_create_node]: (INFO) [/hello_world/vagrant-ubuntu-trusty-64]
```

## 1.2.2 Commandline Execution

Assuming that you've followed the installation guide and installed `zk_watcher` locally, you can run it on the commandline with the following arguments.

```
# zk_watcher --help
Usage: zk_watcher <options>

Options:
  --version          show program's version number and exit
  -h, --help        show this help message and exit
  -c CONFIG, --config=CONFIG
                    override the default config file (/etc/zk/config.cfg)
  -s SERVER, --server=SERVER
                    server address (default: localhost:2181)
  -v, --verbose     verbose mode
  -l, --syslog      log to syslog
```

## Service Configs

The service itself reads in a configuration file (`-c`) that is filled with sections. Each section represents a single unique path that `zk_watcher` will register a node at, and the corresponding check information.

A configuration file that checks two different services could look like this:

```
[ssh]
cmd: /etc/init.d/sshd status
refresh: 60
service_port: 22
service_hostname: 123.234.123.123
zookeeper_path: /services/ssh
zookeeper_data: { "foo": "bar", "bah": "humbug" }

[apache]
cmd: /etc/init.d/apache status
refresh: 60
service_port: 22
zookeeper_path: /services/web
zookeeper_data: foo=bar, bah=humbug
```

### 1.3 Authentication

If you wish to create a Digset authentication token and use that for your client session with Zookeeper, you can add the settings to the config file like this

```
[auth]
user: username
password: 123456
```

If you do this, please look at the `nd_service_registry` docs to understand how the auth token is used, and what permissions are setup by default.



---

## Indices and tables

---

- `genindex`
  - `modindex`
  - `search`
-