# Deployment Manual

**Part #1 - Installation:**

**Tested on Ubuntu 14.04 (most unix systems with docker support will also work)**

**1.1**

```
# sudo apt-get install puppetmaster
```

**1.2**

```
# puppet module install garethr/docker
```

**1.3**

```
# put site.pp in /etc/puppet/manifest/
```

**1.4**

```
# puppet apply /etc/puppet/manifest/site.pp
```

**1.5**

```
# stack is ready for usage, icinga2 post-configuration with slaves.
```

**Part #2 - Deployment**

So, I’ve created 2 images for db and web, and used icinga docker.
My docker account is lexluter1988/

```
# cat /etc/puppet/manifests/site.pp
```

```
class { 'docker':
  manage_kernel => false,
  version => 'latest',
}

docker_network { 'crossover-net':
  ensure   => present,
  driver   => 'bridge',
}

docker::image { 'lexluter1988/httpd':
  image_tag => 'centos6'
}

docker::image { 'lexluter1988/mysqld':
  image_tag => 'centos6'
}

docker::image { 'jordan/icinga2':
  image_tag => 'latest'
}

docker::run { 'webserver':
  image        => 'lexluter1988/httpd:centos6',
  net          => 'crossover-net',
  ports        => ['80:80'],
  hostname     => 'web.crossover-net',
  dns          => ['8.8.8.8', '8.8.4.4'],
}

docker::run { 'mysqlserver':
  image        => 'lexluter1988/mysqld:centos6',
  net          => 'crossover-net',
  ports        => ['3306:3306'],
  hostname     => 'mysql.crossover-net',
  dns          => ['8.8.8.8', '8.8.4.4'],
}

docker::run { 'icinga':
  image        => 'jordan/icinga2',
  net          => 'crossover-net',
  ports        => ['3080:80'],
  hostname     => 'icinga.crossover-net',
  dns          => ['8.8.8.8', '8.8.4.4'],
}
```

That manifest file is really simple, icinga web server will not override 80 port of Apache and all services will work.
My Dockerfile-s just added ‘boto3’ support, ‘backup’ scripts and some hooks like ‘epel’ repo installation.
Centos 6 is used as most common enterprise rpm-based server. Most popular for me, honestly.

**Part #3 - Usage**

Stack will provide us with 3 docker containers on same net, all can see in same net.

```
root@lexluter1988-SVF1532P1RB:~# docker ps
CONTAINER ID        IMAGE                           COMMAND
CREATED             STATUS              PORTS
NAMES
4e080e095531        lexluter1988/httpd:centos6      "/run-httpd.sh"          16
seconds ago         Up 15 seconds       0.0.0.0:80->80/tcp
webserver
37d371a48aeb        jordan/icinga2                  "/opt/run"               16
seconds ago         Up 16 seconds       443/tcp,   5665/tcp,   0.0.0.0:3080->80/tcp
icinga
8b65c3267eb0        lexluter1988/mysqld:centos6      "/bin/bash  /start.sh"     17
seconds ago         Up 16 seconds       0.0.0.0:3306->3306/tcp
mysqlserver
```

Both MySQL and Apache, have centos6 icinga2 service installed.
I do not competed fully-automated assignment of nodes into icinga2 master node.
Therefore both hosts should be configured via ‘icinga2 node wizard’ or via guide

**http://serverfault.com/questions/647805/how-to-set-up-icinga2-remote-client-without-using-cli-wizard**

**Part #4 - Assumptions**

**4.1** Host puppetmaster system is physical host or fully virtualized VM, Ubuntu 14.04. Puppetmaster installed via apt.
**4.2** Icinga installed from pre-configured icinga2 docker image.
**4.3** Both MySQL and Apache will be Centos6 x86_64 based
**4.4** AWS S3 Account details (key) is hardcoded in simple python script, only for uploading via crond every day at 7PM
**4.5** Initally, bash script would do preparation and compress logs/backup, and will call aws.py with boto3 module just for uploading.
**4.6** MySQL and Apache docker images were taken from official community, changed, added in my personal repo.
I added epel repo, pip installation, icinga2 installation, added crond installation, initialization and backup/upload scripts.

**Part #5 - Documentation of issues faced during task**

**5.1** https://docs.docker.com/engine/admin/puppet/
Puppet with docker, manifests, how configure full-stack, network, etc

**5.2** http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html and http://boto3.readthedocs.io/en/latest/guide/migrations3.html#creating-the-connection
AWS S3 has issue that bucket should be unique, as well as in python you should catch case where bucket already created.
Also, with regions, it is better to create bucket with defined region.

**5.3**
http://docs.icinga.org/icinga2/latest/doc/module/icinga2/chapter/icinga2-client
Icinga2 wizard, loong configuration.

**Part #6 - Instructions to prepare code**
**6.1**

Both docker sources are fork from official with some hooks.

```
3 [error opening dir]
httpd
├── aws.py
├── backup_webserver.sh
├── Dockerfile
└── run-httpd.sh
mysql
├── aws.py
├── backup_dbserver.sh
├── config_mysql.sh
├── Dockerfile
├── mysql_build.log
├── start.sh
└── supervisord.conf
```

**6.2** Script for upload could be automated, to get key from ENV or from encrypted custom values, you must have boto3 and python anyway.

```
import boto3
from sys import argv

# Let's use Amazon S3
s3 = boto3.resource(
       's3',
       aws_access_key_id='AKIAJK3EYZYK***JQ',
       aws_secret_access_key='sUURZX0YvPf/F+UnZrGv0xchMW****MLeQ',
       )
try:
    s3.create_bucket(Bucket='web-apache-bucket', CreateBucketConfiguration={
              'LocationConstraint': 'us-west-1'})

except:
    # assuming folder already created
    pass

file = argv[1]

s3.Object('web-apache-bucket', file ).put(Body=open( file, 'rb'))
```

**6.3** Images could be build-ed easily like any other docker image

```
docker build -t lexluter1988/httpd:centos6 .
```

**6.4** Additional actions, default centos6 container does not have
'crontabs' rpm. So, no jobs will work. Icinga2 for Centos6 is also not publicly available.
As well as pip, only with epel repo.

```
# Installing crontabs, since base centos6 Docker  does not have it
RUN yum -y install crontabs
RUN chkconfig crond on
RUN service crond start

# Installing additional modules for python and AWS

RUN rpm -ivh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
RUN yum -y install python-pip
RUN pip install boto3

# Now we add our script to run at 7PM every day

ADD backup_webserver.sh /backup_webserver.sh
ADD aws.py /aws.py
RUN echo '0 19 * * * /backup_webserver.sh' >> /etc/crontab
RUN service crond start

# Now we install icinga2 Agent, since that will be our slave host

RUN                                       rpm                                       -i
http://packages.icinga.org/epel/6/release/noarch/icinga-rpm-release-6-1.el6.noarch.rpm
RUN yum -y install icinga2
```

### 6.5 backup_*.sh

That script is invoked into crond on 7pm every day.
Workflow is
-> # prepare folders
-> # copy logs and backups to temp folders
-> # tar.gz them with timestamp
-> # invoke aws.py from bash with filenames.

That’s it.
No hard issues actually, task is very clear. Icinga hosts registration doesn’t automated by me. That was the only issue.