TUTORIAL: Fast Web API Deployment on Windows Azure

Because APIs are platform agnostic they can be deployed on any platform. In the past we've covered tutorials on Heroku and Amazon deployments.

Today's tutorial is fast web API deployment on Windows Azure. We will use Ruby Grape gem to create the API interface, an NGINX proxy, Thin server and Capistrano to deploy using command line.

For the purpose of this tutorial you can use any Ruby based API running on Thin server. Or you can clone our SentimentAPI.

Locally run git clone git@github.com:picsoung/sentimentAPI.git cd sentimentAPI/

Creating and configure Windows Azure VM

We will start to generate a *X509 certificate with a 2048-bit RSA keypair* to ssh into your Azure VM. It will be useful when you will setup your VM.

To generate this type of key you can run the following command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout myPrivateKey.key -out myCer
t.pem
```

Now, let's get started by creating your Windows Azure account. For this tutorial you can use the Free Trial option. Once the Azure account is created, go to the Dashboard on the Virtual Machines tab. There, you will be guided to create your first VM. Choose the *From Gallery* option, select an *Ubuntu Server 12.04 LTS*.

On step 2 you will be able to upload the pem you created earlier, you should not be prompted for your password again.

On steps 3 and 4, choose the options that suit your needs best.

It will take a couple of minutes until your VM is ready. When it is you will be able to access it's dashboard where you can monitor the activity (CPU, disk , network) of your VM, and upgrade it's size.

The VM comes with few packages installed, so we'll need to access it to install other components. Once the key is created you can ssh to your VM

```
ssh -i myPrivateKey.key -p 22 username@servicename.cloudapp.net
```

Once in it the VM, run the following commands to install all we need

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y install ruby1.9.3 build-essential libsqlite3-dev libpcre3 libpcre3-dev
 libssl-dev openssl libreadline6 libreadline6-dev libxml2-dev libxslt1-dev
```

You can check that Ruby installation is complete by running

```
ruby -v
```

it should output something like *ruby 1.9.3p194 (2012-04-20 revision 35410) [x86_64-linux]*.

we also need to install `bundler` and `thin`:

```
sudo gem install bundler
sudo gem install thin
```

Now, we should have all we need on the VM. Go back to the VM's dashboard and click on the *Endpoints* tab. There, add the `HTTP` endpoint on port `80`, and the fields should auto-fill.

Install OpenResty

In order to streamline this step we recommend that you install the fantastic `OpenResty` web application which is the standard NGINX core bundled with almost all the necessary 3rd party NGINX modules built-in.

On your Azure VM
Compile and install NGINX:

```
cd ~
sudo wget http://agentzh.org/misc/nginx/ngx_openresty-1.5.8.1.tar.gz
sudo tar -zxvf ngx_openresty-1.5.8.1.tar.gz
cd ngx_openresty-1.5.8.1/
sudo ./configure --prefix=/opt/openresty --with-luajit --with-http_iconv_module -j2
sudo make
sudo make install
```

Configure your github repo

In this Windows Azure API tutorial we use github to host our code. If you don't already have a repo for your API, make sure to create one and host it on github.com. If you are not familiar with git and github you can check a great tutorial here.

To use git on your VM and have access to your github repo you need to generate an SSH key on your VM and add it to Github as explained here.

WARNING: Hosting your code on a public Github repo makes it vulnerable. Make sure it does not contain any sensitive information such as provider keys before push it publicly.

We won't deal with github anymore.

Configure your API

How the system will work :

1. Thin server will be launched on port 8000
2. the upstream `YOURAPINAME` is listening on localhost:8000
3. Upcoming connection on port 80 (as defined on the `server` section) are "redirected" to `YOURAPINAME`

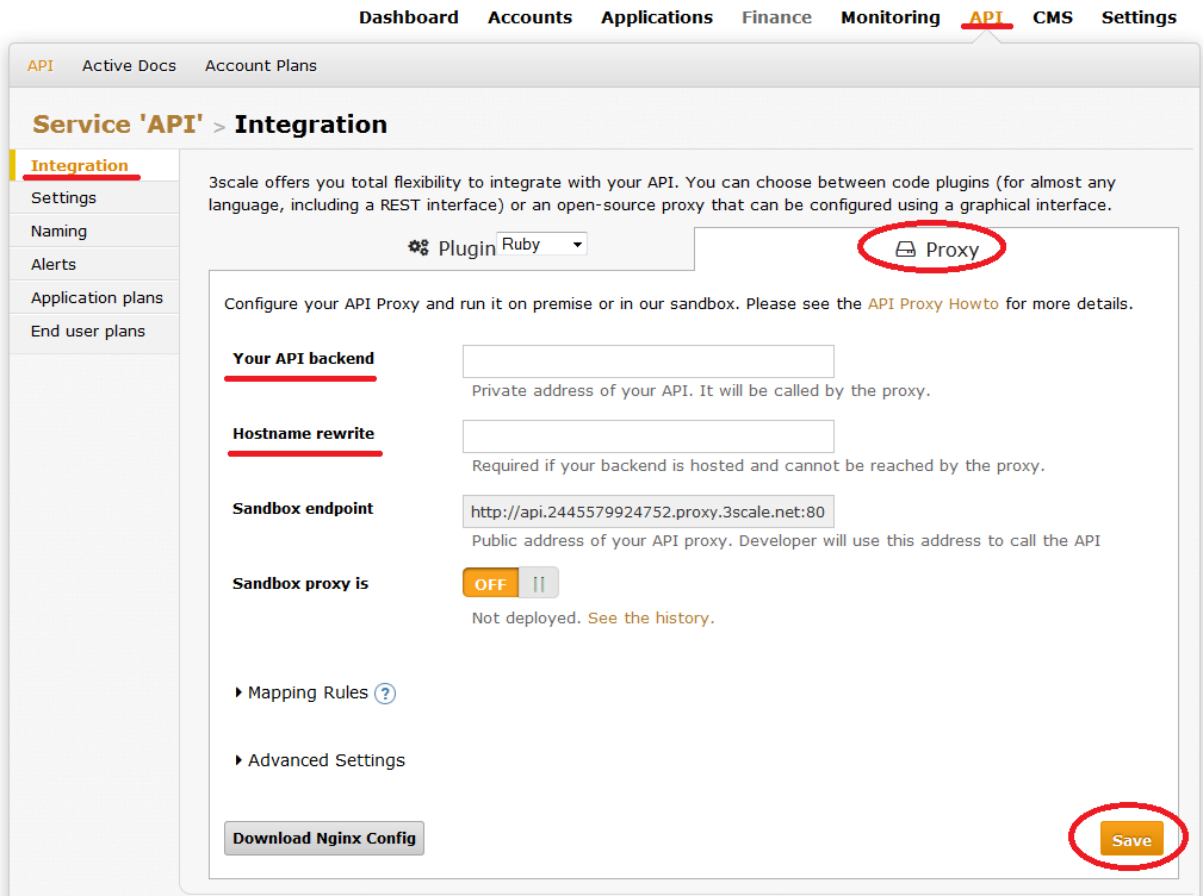
On 3scale

Rather than reinvent the wheel and implement rate limits, access controls and analytics from scratch, we will leverage the handy 3scale API Management Service. Get your free 3scale account, activate and log-in to the new instance through the provided links. The first time you log-in you can choose the option for some sample data to be created, so you'll have some API keys to use later. Next you should go through the tour to get a glimpse of the systems functionality (optional) and then go ahead with implementation.

To get some instant results we'll start with the sandbox proxy which can be used while in development. Then we will also configure an NGINX proxy which can scale up for full production deployments.

There is some documentation on the configuration of the API proxy at 3scale: here and for more advanced configuration options here.

Once you sign into your 3scale account, launch your API on the main Dashboard screen or Go to API->Select the service (API)->Integration in the sidebar->Proxy



Set the address of your API backend – `http://YOURAPP.cloudapp.net:80` . Now you can save and turn on the sandbox proxy to test your API by hitting the sandbox endpoint (after creating some app credentials in 3scale): `http://sandbox-endpoint/v1/words/awesome.json?app_id=APP_ID&app_key=APP_KEY` where, `APP_ID` and `APP_KEY` are id and key of one of the sample applications which you created when you first logged into your 3scale account (if you missed that step just create a developer account and an application within that account).

Try it without app credentials, next with incorrect credentials, and then, once authenticated within and over any rate limits that you have defined. Only once it is working to your satisfaction do you need to download the config files for NGINX.

Note: any time you have errors check whether you can access the API directly: your-public-dns:3000/v1/words/awesome.json. If that is not available, then you need to check if the AWS instance is running and if the Thin Server is running on the instance.

There you will be able to change your API backend address to `http://YOURAPP.cloudapp.net:80`

Once you are done, you can click on `Download your nginx config`. That will download an archive containing a `.conf` and `.lua` file we are going to use to configure your app.

Modify the `.conf` accordingly :

If the API gateway and the API are on the same VM. delete the block

```
upstream backend_YOURAPP.cloudapp.net{
server ....
}
```

and replace it by

```
upstream YOURAPINAME {
server 127.0.0.1:8000;
}
```

WARNING: YOURAPINAME can only contain URL valid characters as defined RFC 3986

in the `.lua` file modify the line `ngx.var.proxy_pass = "http://backend_YOURAPP.cloudapp.net"`
by `ngx.var.proxy_pass = "http://YOURAPINAME"`

In all cases.

Replace `server_name api.2445580546262.proxy.3scale.net;` by
`server_name YOURSERVICENAME.cloudapp.net;`

in `server` block add on top

```
root /home/USERNAME/apps/YOURAPINAME/current;
access_log /home/USERNAME/apps/YOURAPINAME/current/log/thin.log;
error_log /home/USERNAME/apps/YOURAPINAME/current/log/error.log;
```

replace `access_by_lua_file lua_tmp.lua;`
by `access_by_lua_file /opt/openresty/nginx/conf/lua_tmp.lua;`

and before `post_action /out_of_band_authrep_action;`
add

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
```

Finally rename those files `nginx.conf` and `tmp_lua.lua`

Capistrano setup

To deploy the API we use `Capistrano`. Capistrano is an automation tool, that will let you setup tasks for your deployments and execute them using a command line interface. Capistrano is used on your local machine to deploy on your remote VM.

To install Capistrano add this line to your Gemfile

```
gem 'capistrano'
```

Run locally, the following command to install the new gems and setup Capistrano

```
bundle
capify .
```

Copy `nginx.conf` and `tmp_lua.lua` into `/config`.

Capistrano setup

When we ran the `capify` command, we created two files, `Capfile` and `deploy.rb`.
In `deploy.rb` you describe all the commands necessary to deploy your app.

In `/config` edit `deploy.rb` and replace the content by the following

```
require "bundler/capistrano"
set :application, "YOURAPINAME"
set :user,"USERNAME"
set :scm, :git
set :repository, "git@github.com:GITHUBUSERNAME/REPO.git"
set :branch, "master"

set :use_sudo, false

server "VNDNSname", :web, :app, :db, primary: true

set :deploy_to, "/home/#{user}/apps/#{application}"
default_run_options[:pty] = true
ssh_options[:forward_agent] = false
ssh_options[:port] = 22
ssh_options[:keys] = ["/PATH/TO/myPrivateKey.key"]

namespace :deploy do
    task :start, :roles =&gt; [:web, :app] do
      run "cd #{deploy_to}/current &amp;&amp; nohup bundle exec thin start -C config/pr
oduction_config.yml -R config.ru"
      sudo "/opt/openresty/nginx/sbin/nginx -p /opt/openresty/nginx/ -c /opt/openresty/
nginx/conf/nginx.conf"
    end

    task :stop, :roles =&gt; [:web, :app] do
      run "kill -QUIT cat /opt/openresty/nginx/logs/nginx.pid"
      run "cd #{deploy_to}/current &amp;&amp; nohup bundle exec thin stop -C config/pro
duction_config.yml -R config.ru"
    end

    task :restart, :roles =&gt; [:web, :app] do
      deploy.stop
      deploy.start
    end

    task :setup_config, roles: :app do
      sudo "ln -nfs #{current_path}/config/nginx.conf /opt/openresty/nginx/conf/nginx.c
onf"
      sudo "ln -nfs #{current_path}/config/lua_tmp.lua /opt/openresty/nginx/conf/lua_tm
p.lua"
      sudo "mkdir -p #{shared_path}/config"
    end
    after "deploy:setup", "deploy:setup_config"
end
```

This will make sure that Capistrano doesn't try to run `rake:migrate` (this is not a Rails project!)

```
task :cold do
  deploy.update
  deploy.start
end
```

In above text, replace the following:
`VNDNSname` with your .cloudapp.net DNS
`YOURAPINAME` with your applicationame

`USERNAME` with the username used to login into the VM
`GITHUBUSERNAME` with your Github username
`REPO` with your Github repo name
`/PATH/TO` with the path to access the SSH key created before

This above works well if you don't have a database in your API. If you do have a database, you can comment the lines:

```
task :cold do
  deploy.update
  deploy.start
end
```

We also need to add a file `production_config.yml` in `/config` to configure the thin server

```
environment: production
chdir: /home/USERNAME/apps/YOURAPINAME/current/
address: 127.0.0.1
user: USERNAME
port: 8000
pid: /home/USERNAME/apps/YOURAPINAME/current/tmp/thin.pid
rackup: /home/USERNAME/apps/YOURAPINAME/current/config.ru
log: /home/USERNAME/apps/YOURAPINAME/current/log/thin.log
max_conns: 1024
timeout: 30
max_persistent_conns: 512
daemonize: true
```

Again, change usernames and paths accordingly.

Commit the changes on the project and upload them on Github

```
git add .
git commit -m "adding config files"
git push
```

We are almost done :)

Deploy

For your local development machine, run the following command to setup the remote Azure VM:

```
cap deploy:setup
```

You should not be prompted for a password if the path to your ssh key is correct.
Capistrano will connect to your VM and create an `apps` directory under the `home` directory of the user account.

Now, you can deploy your API to the VM and launch thin server using the command:

```
cap deploy:cold
```

This command should get the latest commit on your github, launch openresty and launch thin server.

Your API should now be available on the url `MYAPI.cloudapp.net/path/to/resources`
In the case you deploy the Sentiment API V2. `MYAPI.cloudapp.net/v2/words/hello.json?app_id=APPID&app_key=APPKEY`

Troubleshoot

If you are not able to access to your API, ssh to your VM and check that you can call it on `localhost` using `curl` . Like this `curl -X GET http://localhost:8000/v2/words/hello.json?app_id=APPID&app_key=APPKEY` If it works, there is something wrong in nginx configuration.

You can check nginx logs on your VM with `cat /opt/openresty/nginx/logs/error.log` .

You should now have an API running on an Azure Linux instance.

Hope you enjoyed this tutorial. Please let us know if you have any questions or comments. We look forward to hearing from you.