Udemy Goodness – A Python Quick Start

Part of what makes being a technologist a challenge is finding the right guidance on tools and technology. There are a lot of resources like blogs and community shared content, and as more education sites pop up, we are also getting the advantage of more shared, freely available content.

I was pointed to Udemy recently for a Python Tutorial which I found to be great. For those who wanted to get a quick start with learning the basis of Python, you can visit here to see a very nice walkthrough of some core concepts in Python and programming in general.

As you may have seen a lot lately, development skills are quickly moving up the "must-have" list for operations and virtualization admins. It is no surprise as more content moves towards the concepts of Infrastructure-as-Code, and the further embracement of DevOps concepts by organizations demands new skills across the board.

If you have any free resources that you enjoyed, please feel free to share them in the comments below.

Happy Learning!

The art of the takeoff: technology blogging for enabling experimentation

In the world of technology, it can never be said that there is a shortage of things to learn. With that said, I wanted to highlight what I find particularly cool about the technology communities and what bloggers can do for the larger audience.

Think Globally, Act Locally: Technology Edition

You've probably heard the phrase "Think Globally, Act Locally" which referred to the idea of thinking of the bigger picture, but taking action in your own nearby ecosystem. The heart of the message was about preserving the greater good by local action. Enabling greater ideas through introducing small local changes.

With what our technology communities are doing these days, we see the same thing. Great authors, architects, thought leaders, and technologists are providing the tools to the greater community. We have the opportunity to see lots of live demos of products that are fully in action, but there is a something missing when we just see the product fully operational but only by seeing someone else doing the driving.

The Art of the Takeoff

I've flown a plane and a helicopter in my life. The disclaimer here, and an important point is that I'm not a pilot. But the point of is that I have taken the stick while in-flight in a Cessna 172 and Bell Helicopter (a 407 I believe) which was actually really easy…and lots of fun!

I could never say that I am a pilot despite operating these big machines during a flight, I didn't do any of the difficult work. Just like getting the live demo of some really cool product though, we can't really say we experienced it until we held the stick ourselves.

The point of what I am saying here is that by providing me with a working model (aka flying object), I was able to take part in the active use of it without going through the really tricky first steps. This is the heart of what we need to do as bloggers and technology leaders: enable experimentation.uct, I won't ever really have a good comprehension of this unless I learn the takeoff. However I was able to try out the process safely without the massive investment up front.

It is all about sharing the important portion of the product or technology, which is the build section. By enabling the reader to get right to the flight portion, we can really increase the chance of adoption of the technology and increase the excitement among the user community.

Enabling by Coaching the First Steps

A great example of what I'm talking about is here in an article penned by Scott Lowe (@Scott_Lowe): http://blog.scottlowe.org/2013/11/25/a-brief-introduction-to-linux-containers-with-lxc which is a nice introduction into using LXC (Linux Containers). The article is succinct, and provides the steps to get an immediate result.

I've spent a significant amount of time working with OpenStack lately, and I can tell you that the takeoff part of the adventure is very certainly the most difficult part. Luckily, we have a lot of great resources to enable those first steps with build scripts, packaged VMs, and much more being shared out in the blogger ecosystem.

Kenneth Hui shared a great article which provides an OpenStack Havana deployment using Vagrant and Chef to deliver the Rackspace Private Cloud edition: http://cloudarchitectmusings.com/2013/12/01/deploy-openstack-havana-on-your-laptop-using-vagrant-and-chef/ and I can tell you that this is a massive time saver.

Getting you to the Start Line

The goal of these "Getting Started" posts is really to get you past the point of yak shaving in order to have an environment to work in and really test drive the technology. Plus, using the same information being shared in the scripts and toolkits that provide these packaged builds, you can reverse engineer a lot of the process to learn how to do the same thing yourself. But the value is immediate because you are able to have a working product with limited intervention.

So your goal should be to pick a topic that may seem slightly out of reach, and do a little bit of searching for someone who has written a "getting started with…" post to save you the first steps. Think of it like training wheels for the first part of your test drive with a product. It's sometimes hard to get the machine rolling, but with the help of the great community contributions by our technology blogger community, we will do what we can to get you underway.

If you have had success with getting some new technology or process working, it is definitely encouraged to share your story and your experiences. This is how we got to where we are today with so many products and processes.

Feel free to reach out to me if you have any technology that you are looking to test drive that I can help with, and I will do whatever I can to connect you with the right resources. In our social ecosystems we have great people who would love to help you out! Make sure to send out a Tweet or +1 to the authors, because the feedback you give provides the fuel for more great info sharing like this in the future.

Puppet 101 – Basic installation for master and agent machines on Ubuntu 12.04 with VMware Workstation

You don't have to go far to hear the word Puppet these days. Configuration management isn't just a new trend that is hitting IT environments. It is a methodology for stateful, repeatable, DevOps style management of your infrastructure.

For the VMware folks, this is now a part of the vCloud ecosystem in a way because of the integration of Puppet into the vCloud Hybrid Service (vCHS) that was recently announced.

Most Common Question

Nearly every single day I hear from someone "Puppet looks cool, I need to figure out how to use it". This inevitably leads to the most common question: "How do I get started with Puppet?"

Before you say "I don't use Linux", don't worry. What I want to do here is to show you the simplest step-by-step way to install Ubuntu on a server and a client with Puppet server and a Puppet agent configuration.

Let me be clear that our Puppet installation here isn't difficult, but it does require some time and you have to be sure to follow the process, albeit concise, without missing any steps.

Two Flavors of Puppet to Start Off With

There are two distinct types of Puppet deployment: Enterprise or Open Source. This leads to the next challenge for many, which is finding out which is the appropriate one to run with. Hint: They are both great to start with

With the **Puppet Enterprise** environment (https://puppetlabs.com/puppet/puppet-enterprise/) you can deploy easily using the web dashboard, manage your environment with the full backing of the Puppet team. It has all sorts of reasons to make it an amazing fit for you.

That being said, it is also a commercial product which comes with a license and support cost. This is not a problem for your real production deployment where you need/want that support, but we just want to kick the tires on it first to figure out how to use it.

Enter **Puppet Open Source** (https://puppetlabs.com/puppet/puppet-open-source/) which lets us use

the fully open source version of the product, and the price tag is a grand total of zero dollars. Well, capital expense is zero, but there are some requirements for running this which include comfort with the Linux environment.

Basic Installation For Brand New Users

I'm going to use my VMware Workstation environment to start things off. I also have the Ubuntu 64-bit Server 12.04 LTS (Long Term Support) ISO file which I'll use for deploying. Before we get started, you need to go here to download that: http://www.ubuntu.com/download/server

Now that you have your requirements, let's install our two virtual guests which will be using DHCP (easiest deployment to get us started quickly) using the names ***puppetserver*** and ***puppetclient*** for our virtual machines. We will use the VMware Workstation "easy install" wizard to quickly deploy the guest OS.

Install Our First VM (puppetserver)

Using or VMware Workstation, we will create a new virtual guest by choosing File | New Virtual Machine and using the steps below:

At this point you have to choose your username that will be your login account to the Ubuntu console. Type in your full name, a user name and a password:

Install Our Second VM (puppetclient)

Rather than repeating all of our screen shots, just repeat the process as you did above, except you need to select puppetclient as the VM name in step 5.

Now that you have deployed your second machine, you can see the two guests in your VMware Workstation window.

Yay! We have our two guest machines ready to start our Puppet deployment.

Configuring our Puppet Master (puppetserver)

Log in to the console using the credentials you defined during the setup wizard. We will launch all commands using the sudo command which will elevate our privileges to launch each action. The first time you use sudo, you will be prompted for your credentials. Further sudo commands in the same session will use those cached credentials.

Update the apt repositories with the apt-get utility

```
sudo apt-get update
```

Change the host name using sed (Linux stream editor)

```
sudo sed -i 's/ubuntu/puppetserver/g' /etc/hostname
```

Set our IP up for the network interface eth0 using nano (a fairly simple Linux editor). Once we edit the info with the format below, using your own IP information, we just type Ctrl-X, then type Y followed by pressing Enter to confirm the file name, save and exit. While I don't personally use nano, the general usage is simple for those just getting started.

```
sudo nano /etc/network/interfaces
```

Add a record to the Hosts file for our server and our client (use your own IP addresses of course)

```
sudo nano /etc/hosts
```

Install Puppet

```
sudo apt-get install -y puppetmaster
```

Stop the puppetmaster service

```
sudo service puppetmaster stop
```

Remove the default certificate

```
sudo rm -r /var/lib/puppet/ssl
```

Reboot the server

```
sudo reboot
```

Log in to the console once the reboot is completed. We will do some tasks after our client is configured.

That’s it. No, really, that’s the end of the server configuration. Next we configure our client machine

Configuring our Puppet Agent (puppetclient)

Now we log in to the console of our second guest which will become the agent managed machine. We will be performing the following steps:

Update the apt repositories with the apt-get utility

```
sudo apt-get update
```

Change the host name using sed (Linux stream editor)

```
sudo sed -i ‘s/ubuntu/puppetclient/g’ /etc/hostname
```

Set our IP up for the network interface eth0 using nano (a fairly simple Linux editor). Once we edit the info with the format below, using your own IP information, we just type Ctrl-X, then type Y followed by pressing Enter to confirm the file name, save and exit. While I don’t personally use nano, the general usage is simple for those just getting started.

```
sudo nano /etc/network/interfaces
```

Reboot to ensure the new IP and hostname take effect

```
sudo reboot
```

Log in to the console again

Add a record to the Hosts file for our server and our client (use your own IP addresses of course)

```
sudo nano /etc/hosts
```

Install Puppet

```
sudo apt-get install -y puppet
```

Add the puppetmaster server entry in the puppet.conf file. We will create an *[agent]* section and put an entry which says *server = puppetmaster*

```
sudo nano /etc/puppet/puppet.conf
```

Set the puppet agent to start automatically using sed

```
sudo sed -i 's/no/yes/g' /etc/default/puppet
```

Restart the puppet agent

```
sudo service puppet restart
```

Now are client is fully configured. Let's go back to the puppetserver console and manage the puppet agent request which will have been created by restarting the services on the client.

On the puppetserver console we list the pending certificates and then we sign the certificate with the requesting name attached. This is the sequence that we use:

```
sudo puppet cert –list
```

```
sudo puppet cert sign agentname
```

And now we have our Puppet server and Puppet client all ready to go!!

We have lots of tasks we can do now, but we start with the basic step of creating our first manifest.

Creating your site.pp file

On your puppetserver you need to create your first manifest. With future articles we will do lots more than this, but the first step is to get your initial file started to test the running of the manifest against your new puppet client.

sudo nano /etc/puppet/manifests/site.pp

Add the following content shown in the image and save the file

Next we can either wait for the manifest to run at the next interval…which we won't do because we are impatient

Log into the puppet client console and initiate the puppet agent by restarting the service

```
sudo service puppet restart
```

Now we will run the puppet agent to test the connection

```
sudo puppet agent –test
```

Yay! It worked. Let's confirm the file has shown up as we defined it in our site.pp manifest:

So now we have successfully completed a few tasks:

1. Installed our puppet server
2. Installed a puppet client
3. Connected the puppet client to the server
4. Created a basic (very, very basic) manifest
5. Run the manifest against our client

This may not seem like a big accomplishment for some, but surprisingly there are many administrators who have little or no experience with using Linux systems, and the first few steps, although simple, can be daunting for some.

In future posts we will take the next important steps of creating manifest modules, deploying the puppet client to a Microsoft Windows machine, and then getting into more detailed manifest configuration.

Puppet Cheat Sheets

We will use these for our next post as we start to build some basic manifests for testing out the various components. We can use these to help us dip our toes further into the water of Puppet automation.

Puppet Labs Module Cheat Sheet – http://docs.puppetlabs.com/module_cheat_sheet.pdf

Puppet Labs Core Types Cheat Sheet
– http://docs.puppetlabs.com/puppet_core_types_cheatsheet.pdf

So read those guides and we will be adding some more flavor to our little Puppet test environment very soon! I hope that this is a helpful guide to get people started on your journey with puppet

Get-Started: ii Captain! Using the Invoke-Item CmdLet

A nifty little PowerShell CmdLet to make use of is **Invoke-Item**, which also has the alias **ii**. This CmdLet can be used against files, URLs and other documents and items to invoke the default action against them.

An example where we would want to use this is where we generate some output into a file such as a .txt file, and then we want it to be rendered to the screen using the editor for the .txt extension which is most likely Notepad.exe on your system.

In this case, let's use a simple one-liner to output the path variable ($env:path) to a text file.

```
$env:path | Out-File MyPath.txt
```

Now that we have the file created, we simply run the Invoke-Item against the file to launch it using the default Open action based on the file type association.

```
Invoke-Item .MyPath.txt
```

The result of the **Invoke-Item** command is that you will see a Notepad windows launch (or whatever your associated application happens to be).

This was a very simple example, but the concept is identical for any registered file type in your system. When running as an interactive script, a very popular purpose for the **Invoke-Item** CmdLet is to launch Internet Explorer to view an HTML file which was created by some other process.

While this is a handy little tool in our PowerShell toolbox, you may find that you want to move to the **Invoke-Expression** if you require more complex command lines and parameters to be available. I'll be putting together a post on that CmdLet in the future which will go into much further detail.

The CmdLet supports relative and literal path references and requires double-quotes for long file and path names where spaces are in the path.

The Get-Help for the Invoke-Item is as follows:

The full TechNet help for the Invoke-Item can be found here:
http://technet.microsoft.com/en-us/library/hh849794.aspx

Happy invoking!