Transcript: Getting Started with DevOps
Free Webinar on June 23, 2014

Subu Sankara: Hello everyone. It's just about time to start the Webinar. Good morning, good afternoon depending on where you are in terms of location. Welcome to Synerzip's technical webinar series. We are going to have a Getting Started with DevOps topics today. As you might well be aware of in the world continuous delivery big data and software as a service they all have one thing in common; they ride on the back of something called DevOps that is a combo word of development plus operations.

DevOps is an essential piece of the puzzle when it comes to achieving faster time to market, more robust deployments, better release and stress free release cycles. In this webinar we are going to look at some key things in continuous delivery which DevOps fulfills to a great extent. We will also see why big data needs DevOps from day one and software as a service needs DevOps from day one.

This webinar will focus on DevOps life-cycle, popular DevOps tools and how they work together. Now I'm going to talk a little bit about the presenter. Our presenter is going to be Rohit Ghatol. He is Director of Engineering at Synerzip working on many projects, he started a lot of initiatives for Synerzip and a technology evangelist and he is an author and a speaker at many technical conferences and he is located in the San Francisco bay area for us, so without much ado I would like to hand it over to Rohit to continue on this webinar.

Some of the logistics feel free to enter your questions whenever you feel like and I will try to schedule the questions in its appropriate logical times in the webinar so that we don't dislodge the flow of the webinar. Without much ado Rohit here it is.

Rohit Ghatol: Thanks, Subu thanks for the introduction. Let's get started, what we are going to talk about today, we are going to talk about the problems which you face and you don't have something like DevOps in place when you are doing big data or continuous delivery or software as a service. We are going to go over that part briefly. We are going to over the part of the culture and the people also briefly because that is a very important part.

Then the rest of the presentation is going to be all about tools, remember this is Getting Started with DevOps so we won't be doing comparison of the tools but this is more like a sharing of our experience of what works

for us. There are many tools out there and I'm sure not all equal, nor covered, but still all good tools. Let's get started with the topic.

Look at this image for a while, in case you are working with software as a service, you'll see some solutions are … Or you are trying to work with the continuous delivery model or decided to work with big data. The one thing which you are going to face is of thinking that delivery is not a very easy thing to implement in your organization and that is what this image signifies.

You will have to work hard to make sure that the two parts of your organization actually comes together and start to work together. That is what this image is all about.

Subu Sankara: Rohit, one quick thing; I just wanted to confirm that we are able to see that data slide on the webinar for all the participants. If you have trouble please let me know as Rohit flips through the slides. Please go ahead Rohit.

Rohit Ghatol: I'm going to slide number three and this slide is about the Product Life Cycle. We are very well aware of this but I'll just go over it briefly, we have a planning phase the product owner envisions the product puts priority on this backlog and then there is a release planning to see what are the things you can incorporate in a release.

It is followed by the development creation cycles in planning the development creation cycles are done you have a final build, good QA on that, you mark that particular build as a release candidate and then you flip it over to the operations side.

Now the operations side when deployed would then operate it and it will also monitor it to see if things are going right, asset is right, the comments are right and so on and so on. Now while this is a very familiar cycle to us the one thing we've observed again and again is this arrow between the release and the deploy. That's the arrow that gives us the maximum pain. I've never seen this to be any less than a few weeks if not months especially if you are not following DevOps in your organization.

This is an area that there are minimum people to look at. Let's just go over what particularly happens when we are going from the release to the deploy cycle. This picture is the one which captures the whole reality what really happens. That same team did an excellent job of easing the

release candidate with a good amount of documentation and then they pass it over to the operations side and the maybe they think done a good job.

On the other hand operations side people think that it is crap. The readme files or documentation files are not that great, they have to go back and front, sometimes they will open bugs. You have to close it, or develop will close it saying that they are mostly configuration or environmental issues. This is what particularly happens and that is one area pain that I am going to talk about more and more in this particular presentation.

One aspect of this thing is on the left hand side you have the develop team - the Dev and QA team - who embrace dynamic things, they will basically keep on changing the software style or the tools or the framework library on the right hand side the operation side of people they like things to be more settled. They like stable team work so they don't really embrace change so much as the left hand side group.

Then what we really have in the middle is this wall. We wall this "wall of confusion." This is the wall which really causes a lot of problems, really if you see the arrow from release of deployment that is where you will find a minimum number of people are actually involved but maximum amount of trouble in that particular areas. This what we call as a "Wall of confusion" and it leads to several problems.

It can lead to problems as much as release failure because things does not work right, I'm very sure if you are in dev environment and your production environments are different and educate on one of those things is not done and if you don't have a staging area or even otherwise if you can have service outages because some things were never encountered.

No matter how much you try hard to basically hold back the arms of the clock, you cannot. No matter how many endless hours the operations guys spend to ensure that things simply run, they do not basically the whole idea is that things happen on the left hand side of the wall are different from things happening on the right hand side of the wall. That is what really causes the pain in this DevOps wall scenario.

I'm going to go a little bit ahead and talk about what DevOps is. To try to give DevOps a kind of definition, then we'll know about the culture and people side of DevOps. So we have a screen to look at what DevOps is

really not. This new role is coming up in the market called DevOps, so do not go and hire a person, with the title DevOps and say that your work is done.

You work is certainly not done, even if you go and hire a person such as DevOps you have to empower him to bring about a DevOps culture in your organization, without that you are not going to achieve continuous delivery or much more robust release cycles it's really not about title. It is about getting a culture and empowering people.

DevOps is certainly not about tools, and in all the conversation which you have with my teams or with the prospects or with the client, the people you're talking to is an engineering type of person, they will jump to the conclusion that okay … yeah we understand what the Ops really is and so tell us about the tools which are required and we'll go ahead and do that.

That particularly happens because developers thinks that they are a super set of everything they can play every role. To an extent yes they can play every role but they are not meant to play every role. Each role requires it's own due diligence, so don't get stuck on develop tools, tools are the least important part in DevOps. But anyway as this is a presentation we'll go with some of the tools for getting you an idea of how things go.

Lastly DevOps is not about process, if you are thinking about some heavy dev process that this has to happen before that, that's not what DevOps is. Avoid that thing, DevOps is more in line with Agile, so that's why we treat it like that. This is my definition of DevOps and I like to have my own definition of DevOps it makes me clarify things much more easily.

DevOps is a culture of trust and collaboration and if people use the right tools for automation to achieve continuous delivery. This definition is a bit narrow but it helps in this particular presentation. In the goal I mention is continuous delivery because it talks much more. It talks about having one click deployment cycles, one click rollbacks and much more robust release cycles and so on …

I wanted to lay emphasis on is this "rule of thumb." We are looking at a slide with "rule of thumb" we are look at a part where we are talking about culture is more important than the individual and the individual is more important than the tools. So don't go and buy some tools from some organization or go purchase some open source tools and say you're done; no …

Don't rely on your operations guy who is very, very tech savvy who can just go and just dumping a lot of things on just him. It is more about culture and it is more about everybody playing their role to achieve continuous delivery, achieve rapid recycles or robust recycles with automation and measurements. So we are going to go over some of these things briefly.

I'll just pause over here and ask Subu if there are any questions so far.

Subu Sankara: No Rohit, I encourage the audience to type in their questions, so feel free to type in any time so that I can sequence them to Rohit at appropriate times. But for now I think you can continue.

Rohit Ghatol: Good, so the first thing you have to understand is what are the goals? Does you team even have a common goal? This was actually a problem I saw with teams not able to implement continuous delivery. I suppose people don't understand what continuous delivery is, so I will encourage you to read about that.

The second thing is people have limited vision in terms of their goals. So if you ask Dev and QA team a success software release; they call their goal. If talk about operations guys; they'll call it whatever I've got; I want to deploy it and run it that's my successful goal. Really the goal of the organization is beyond that. The goal of the organization is to get these new features to the end customers and ensure that they are benefitting from those new features.

Then measuring which features are useful to what degree and so on, tweaking the product offering, so as to take it forward. Please ensure that people have common goals.

Subu Sankara: Rohit have our first question from the audience. Can you tell us how do we achieve this common goal?

Rohit Ghatol: Achieving this goal … What typically happens is people don't think about release. Release is a very painful activity for them; they tend to avoid it. This is more true in bigger organizations than in smaller organizations. What we really did, we offered an experiment in slightly mid-size to larger organization is that we put in releases and what we said was, they end of the releases is that they should be a staging environment always.

We are treating it or encouraging people to see a staging in mind if it is always live. What we do is always applied over here in a documented

manner more frequently and that is one step closer to doing that for production. Having more releases is doing something which is more painful more often to figure out easier means of doing that and always having a staging environment over these scenarios. That worked very well for us.

Subu Sankara: Very good and a related question Rohit; does DevOps follow ITIL process?

Rohit Ghatol: Well I don't know really about ITIL process but like I said, DevOps is more about culture people and then tools. Don't compare it to something like process or things like that. There will be aspects of any operations processes which will be common to DevOps and ITIL, but really whatever suits your organization needs, which it then will, that's what you need to do basically. It is more like Agile, Agile is not a strict process but a set of principals.

Subu Sankara: Very good, please proceed.

Rohit Ghatol: The next thing we need to ask is? How often do you actually facilitate a regular depth in operations collaborations? If you are already meeting at the recycle, if these two teams are only meeting at the recycle, then you are going to have issues. Then you cannot avoid delays, you cannot avoid back and front; you cannot avoid operations people evading bugs or shooting emails or crying out loud … If you had to find opportunities, meet more often.

Release timing is a good area, sprint cycles are a good time to meet. I typically encourage cross pollination in my teams. I am never shy to include an IT team person into my sprint cycles or even then a developer or a QA, or IT in the sprint cycles. Be sure that this is done more and more often.

One more thing which you have to understand is the non-functional requirements. This is often a cry area for a lot of people, if you really want software as a service that so many non-function requirements are not talked about in your Agile cycles. Agile will talk about your product features, your user stories but it tends to put this non-function requirements as part of stories, sometimes that is not enough.

You have to find someone in your organization to take care of this non-function in advance, if you want to have a environment in-house, if you want to have a AWS staging environment, a production environment, you have to spend enough time/delivering planning and delivery expense to

achieve to achieve that particular thing, so time finding of incorporating that.

The next thing is very simple, just list things which are not automated/repeatable and document it, and typically if you are doing CID cycles it's easier said than done, you are all doing the automated build, you will testing automated board analysis. All these reports are coming to you.

We need to think to do a step up and talk about the deployments, talking about staging setup, talk about automated rollbacks. You would really have to do all these things to reach there and this is something which is done more and more commonly these days. More importantly you have to talk about things which you are not mentioning today.

Operations guidelines would say; hey we do mention downtime and everything and the performance, that's good, you need to do that. The moment you start to incorporate DevOps or continuous delivery in your organization you have to start measuring some other things. Which is adherence to release dates, release production time and the kind of support defects …? What kind of support defects are you really getting? Are the numbers going down? So not only the aspect of production but aspect process you have to measure.

Things on the right hand side are kind of by-products of calling a develop cycle in the organization or continuous delivery in your organization. That briefly covers our first two topics. What happens when the DevOps is not there and what is the culture and the people part of DevOps? Now we focus more towards the technicality of DevOps and I want you guys to ask some questions on this area because things are fresh in our mind …

Any questions I welcome them right now.

Subu Sankara: I think for now we are good Rohit, please continue.

Rohit Ghatol: We are going to look at the tools box … Typically I would categorize DevOps Tools in these categories. This is my own, mind map of DevOps Tools; I find it easier to think of DevOps Tools in this particular manner. The assumption is that you do continuous integration testing in your organization. In doing that you already have a bit of repository, you already have nightly builds … Tools like "Jenkins" you have automated code quality and assist tools, and the list goes on.

Once you have all those things in place then you start talking about these tools, let's begin with our first thing, we'll talk about machine provisioning a little bit. Before we have that provisioning in software provisioning I'd like to set a few rules which I tend to follow and I benefited from. Number one is: run all your software on production like environment and whenever I say this statement I see how it goes in this phase, I see how is that possible? Well you have to start taking baby steps.

Now ask yourself a question? The developers what do they work on; they either work on Window or they work on Mac? But what's the production environment, specifically Linux and UNIX environment which is there. The first thing is your OS's development you're testing at times and your staging production are different, so that is rule number one. All software run on production like environment, at least start to the same OS if possible and you will see the benefits sometime and the same of setup.

All environments provision using automation. Do not do things by hand and this is more true when doing big data, try to do big data instead of using machine for hoodop tool, you have to install it yourself and try to go back and visit it after two weeks, you'll find that you have to change it because, no other software version is the same.

Difficult environments like Hoodop Tool are crazy environments, there are so many moving pieces with versions changing every second week. How do you keep up with that? You cannot just do it manually or you cannot have images; or you cannot have a ReadMe file. That's not clearly possible. But the thing is you have to have central repository for all provisioning automations and please, please apply this from your first string.

Let's just go to a scenario of what really happens. Let's say developers are working in their Windows boxes, they are testing, they are basically prepping products on something like a LAN stack or a JEE stack, all big data clustering. That typically uses a Windows box a single machine environment, everything working on this one box.

The QA might be doing a better job, they might actually have servers, some of them might be meeting what actually happens in production; but may or may not happen depending on the level of your QA cycles … The impressions obviously will go on a Cloud or they would go on a private virtualization tool and then they will do things.

The typical problem that happens is you have to make sure that things are common across the board, at least developers are using a cut down version of the production environment that would be ideal. If the production environment is having 20 odd servers, let development environment at least have three servers, maybe one for the web server, one for the QAs and one for the Developers, at least lets just have that and just start taking steps from that side.

So Vagrant has set a tool which helps us really well in set up. What Vagrant does is it has its providers for various things like: what your box is beyond there AWS. What it does it just provisions machines and that's what it does. It gives you a machine based on some configurations like; what is the OS? Is it a Cent OS or RedHat? What kind of brand are we looking at? What kind of configuration are we looking at and it does it so.

Once the machines are setup it basically fire scripts like Puppet or Chef to basically provision software. We'll talk about puppet and Chef in a few more slides to come. The most important part of Vagrant is it makes life easier for the developers. If you can feel, that okay are you asking developers to leave their Windows and Mac environments and go with Linux, well definitely not.

With Vagrant what happens is the virtual box, you get two things to do with them. You get assisted script, so you can always assist in your Linux boxes which are a kind of production like environment and then some commands over there. The second thing we get are the Sync Folders, so the Windows boxes and the Linux boxes they get some folders where they can share files.

Imagine a developer using Eclipse on Windows, JEE work file and dumping one of those sync folders and that in the end is a "tomcat's" RedHat folder and things start to run. So, developers having the best of both worlds, they would use development tools on their Windows box, or Mac box and use their own pickup tools with their own browsers and developing tool.

So, really it's a very nice thing to have and going to Vagrant is the first thing which I recommend to everybody. You will see a short demo of that and a few more things.

Subu Sankara: There are a couple of questions if you can take? Would this DevOps be delivered only for software as a service model development or would it makes sense for the conventional packaged software also?

Rohit Ghatol: It can be basically typically at the end, their memory software runs on. You have to make sure that you are running the software on that environment sooner than later. So it's not really about software as a service, but look at your client cases; if you can duplicate your clients environment continuously adhere on site, then you are benefitting from that.

Subu Sankara: Very good, and another question related to Vagrant is how much of challenge it is to make this work on say AWS instance for example.

Rohit Ghatol: Vagrant typically has Plug-ins and these providers which I am talking about are really Plug-ins. So Vagrant is like an abstraction there, you just define your OS, your machine attributes and you run them to this levels; like you have a plug-In which it comes inherently with. You have AWS Plug-In, so you install the AWS Plug-In and you provide Vagrant with the AWS keys and you can do the same on AWS what it does on Virtual Box.

Subu Sankara: Okay, all right please proceed.

Rohit Ghatol: This a typical Vagrant file, I'll quickly run through it. A Vagrant is typically a Ruby file and guys don't be scared, because I don't even know Ruby a lot. Number one it is Ruby, number two it's English so we should be able to know what we are doing. Number two is talking about VIX box, so any virtualization tools starts with some images. Here they are specifying the image name and the name of the image maybe trying to install it from.

Number three is the network attributes, here we are telling what is the host name and what is the IP address; so really it is very, very easy. Number four there is maybe the same, I've done with setting up the machine now I want a tool made perfect to go and install software on that particular machine. So really it is not too difficult even though I'm not a Ruby expert, I can always go and modify these text files.

The next thing we will go and visit is software provision and then we will look at contribution management and quickly we will go over a demo of these two things. Then you will understand how these two things work together.

We are talking about software provisioning, really the nature of this is and I'm talking about Puppet and Chef in general taking them as a context. Is that they are OS that you have, they are RedHat OS, 0.12 OS; Soothsayer or Windows and really you want your tools to move across them. You want to set up your LAN Player, set up on a Windows box or a

Soothsayer.es or a RedHat.es but you don't want to do that over and over again.

You don't want to write those scripts differently for all those things, so really what happens in the software provisioning tools is there are building blocks. These building blocks for example you want to install Java would be first of all to install Java using apt-get on Win tool or Gem on Cent OS and things like that. So really they are building blocks like that. Then you might want to set up soft links to Java and Java C and user-bin and those are also building blocks.

Now, if you take those building blocks together, you take the building blocks installing Java and then linking the Java and Java C files, combine them together and then you get higher level abstract we'll just call this Java in this case together. So Java means doing building block steps in a row and that is what we are talking about high abstract.

Then you definitely want to take out the part where you talk about which version of Java, which is a part of Java and so on and so on. You want to put them in a configuration, so you then change them. You talk about modules with injected configuration as the highest of artifacts. You are talking about software provisioning. These artifacts are stored in your master level, most service tools can run alone, or you can have an agent under a node to read this configuration for these modules install as a part of the software.

They also run in a master/slave manner in which you have a master level, where you think okay, I want to install Java, I want to install Tomcat, I want to install MySQL, I want to install QA software and really what the nodes do is they just … The agent sitting on a node they basically talk to this master level. Then there is a comparison, so agents which run on this, they know about actual state of the machine, every 30 minutes or so they will contact the master level, they'll get a desired state and if there is a difference they will try to patch it up.

This is exactly what happens in Puppet and in Puppet you call the lowest level construct as resources, one example of resources mentioned in this case we are talking about an ATC OS file. We want to make sure that that file exists and that this IP address means this this particular hosting. Finally that is the file security mode.

Now if you combine all those resources together you get something called a CLASS and if you take a CLASS with configuration, you get module

then you get Puppet master and Agents. So really that's how Puppet really works, it is very easy to understand, just look at resources, classes and modules step-by-step and all these things.

Here is an example, a short example of how things really look like, let me give you an idea of what we are talking about over here. Let's look at the class Java, that's so you have so you are declaring installing Java really is … Take a package of Java which would be easy to get on Win2 or it would be Gem on Center West and then try to create a Java link.

When you combine these two resources package and file together you get a construct class Java, similarly for Hadoop it means that I'm going to download a particular file and extract it. When you want to talk at levels of modules, you would say; include Java; include Hadoop and if you look at the file section over here we have a module with two folders, so modules is nothing but a collection of classes and required files.

The site of PT is really the main file of Puppet which says that it's mostly is Hadoop then I want to include a Java model in it and I want to include an Hadoop model in it. Mainly the site of pp will have a number of nodes or number of machines so you can really control which machine gets what modules installed and these modules are a collection of classes as we saw over here and a collection of resources.

Subu Sankara: Rohit will it be a good time to take a question?

Rohit Ghatol: Sure.

Subu Sankara: Thanks, so do we have write these modules from scratch or is there support available especially from open source, if the modules are taken from open source if there is any support?

Rohit Ghatol: The good thing is there is enough support, in fact there is something called Puppet Forge, if Google it and this is only for Puppet and you looked for let's say "Lamp Stack" you get many modules which are there, so this is like source for Puppet, where you pick one and look at those things and you can actually look what these files are, so you get the Puppet files for doing those things.

Let's get back to the presentation and I hope that answers it.

Subu Sankara: Sure so the help is available is what we conclude.

Rohit Ghatol: Yes enough. We've got many tools apart from Puppet and Chef is the one that comes closest to Puppet. It is also Ruby-based like Puppet, Puppet typically if you saw this over here and it's very easy understand this language, this is really like a adjacent like language, we just declare things. Chef on the other hand is based on Ruby so people who are more well versed with Ruby, they like Chef better. They're constructurally the same and there is a good memory support for both Puppet and Chef.

About other tools there are plenty of tools available and we really do need to understand them to great degree to compare them. So here is a good video which you can go in your own time to see which tool actually helps you out better.

What we are going to do next is we're going to go over the configuration management site and then we will go with the demo so that you understand these tools working together in a better fashion. So configuration management, this is nothing new, this has always been there, using software all the time. Think about your Java code, Python code and Ruby code and think about properties file with the constants.

Here also what we've done is this is a Puppet example with Hiera in use a configuration tool. Here we are talking about installing Java and then getting a link at a particular part. If you look at this part Java, that's here … this value is kept in a file called common.YAMLfile, the Java part is really user led YAML they call Java and you want to inject that thing at runtime over here.

This is what we clearly need to take out things, so you are putting place holders for data in your actual code for software provisioning. Now it's not as simple as that when you're talking about handling dozens/hundreds of machine on a software farm because you really want to override these values depending on what the machine hosting is or something that's … if it's a setting environment. Whether it is a production environment, you really want to control these values.

For example over here, if you want to install Java at this particular part on all the machines, but let's say when it comes to a machine named GetO1 example.com, you want to change that and you want to pick it up from different files instead of common.YAML file. So here is an example where we are talking about hierarchies.

The hierarchy really means that if you don't really find the value anywhere else, you get from the common.YAML file otherwise if you find

it somewhere else before that, for example the clients that were here really means the domain name, the hosting. If there is a file called as bev01example.com on YAML, it will try to find that value over there and then it will try to find the value in the common.YAML.

YAML is just another file format, I think it is very easy to understand, the whole idea is let's say you are putting Hadoop in the picture something's in Hadoop they work in a master and slave manner and you really want to go and tweak a value in a master. You create a file with the master.YAML link and you click the values over there. This picks up things over there.

Without further delay let us look at a code example, so things are clearer to us much more. If you going to look at this example, we going to take Vagrant, Puppet and Hiera. We are going to run Vagrant command called this Vagrant-pup which would read a file call this Vagrant file. The Vagrant file will actually have instruction to create a center box on virtual box, once it has done that it will delegate it to Puppet, Puppet will go and read the side door PP file.

It will look up the Java model and Hadoop model, we'll see that there are configure sheet data, to be put up, it will do that and then it will install those things on the provision box. Let's take a look at … So this is the box example, this is our Vagrant file we are talking really about a CentOS box whose URL is this. We are talking about a box being named Hadoop, and with IP address of this.

Once the Hadoop box is provisioned with CentOS and put in the hosting and IP address we are asking Puppet to take control and to basically start provisioning. Puppet typically looks at a site that PP file in manifest folder, maybe just talking about; if the machine name is Hadoop, please include Java and please include Hadoop on that.

These two folders over here named Java and Hadoop and in Java you will see that we have a class in Java and we've got a package which is going to be ?? and then we are trying to create some links for Java C and JPS. For Hadoop it is the number of things including downloading a software package, for Hadoop and then making sure that some configuration files are copied over and finally it is going to start things like a main node, the data node, the resource manager and the manager.

The one thing which you can observe over here is that the user file is called Aspa PP where we talked about which version of JJK you want to

use. This is particularly being read from something called a high rise YAML file. So here we are talking about a hierarchy where we are talking about common.YAML which you will see over here and we are talking about client serve, which means the name of the machine.YAML.

If you go into common.YAML you will see that all the properties are mentioned over here and the OpenJDK version is 1.6 but we just went and overwrote that version to .7 for Hadoop and that's how things really work out. Let's just look at what actually happens, so we are going to into the direct DevOps example and we actually just need a click, and you will see that we've got a Vagrant panel up there.

All we have to do is run a command called S-Vagrant, right now also has one machine running which is my other machine which I will talk about. Here I just say Vagrantup and when I say Vagrantup it is going to be the Vagrant file and it is going to ascend to OS box; likely it is trying to provision the machine.

You will see over here that that machine is coming up … Once that machine comes up it will handle the control, when it's suppose to assist to that machine, now that machine is loading up over here. The moment it connected to that machine it will basically hand over control to Puppet. The machine booted and ready …

It is going to run all the commands in Puppet, right now it is trying to install Java, this might take a while … Once that is done it will go ahead and download the Hadoop file and unzip it put some configuration files in that and five commands to start the main data node and so on … We are talking about this cycle currently going on …

This is doing Java installation, once that is done it goes to the Hadoop installation. Sometimes it takes longer because of network issues but I'll just stay here for a while then … Okay that's done, it is able to download, it is currently downloading my Hadoop package, and it is doing it on a local machine, so that's running faster.

Subu Sankara: While the demo is going on one question has come up. That is apart from Nagios what are the other packages that this would work with?

Rohit Ghatol: Nagios comes later on actually, so Puppet would work with anything actually, so this is really software provision and Nagios is really a monitoring tool which I have really not come to, so let's just wait for a while then we will go to that slide … Here we are just starting up and we

should be done in a moment … The way to verify this thing is … The IP address is, this is to verify that the YAML which we install on this thing is running and this is to install the HTFS on this machine is running, so it is running and YAML is also running. The way I mention for developers is that if they really want to do this, they only had to do this and then they actually go inside that machine. Right now I am inside my Hadoop machine.

Over here a folder this folder is really the folder which is kind of the shared folder, so if I am supposed to go in … over here look at what we have over here, this is the root folder is shared, location/Vagrant. If I can create a file over here, you can see that file is also reflected on this site. That file come under Mac OS over here, this is the same folder …

These two things makes it very easy for developers to work on production-type environment, but not with HTH tools are browsers and things and so on. That was a memo for Vagrant, Puppet and Hiera, if there are any questions around that I would love to take those questions.

Subu Sankara: No, at this time I am just looking at the list of questions, there is nothing related to this, you already mentioned that there is enough community support and all that, where we can get this modules, so I think we are good for now.

Rohit Ghatol: So the next thing is the monitoring tools and there are many tools in this area. Really this is an operations heavy area instead of me answering, it's good to get somebody from our operations side to answer those questions. Many of these tools work in a very simple manner. There is a monitoring tool and there are agents on machines, the agents are using simple tools like SNMB or some other tools.

But there are active checks and passive checks that are happening, active checks is periodic check, if you see a ping on the machine, through http, telnet … That is support that is what we do to figure that that software is running or not. There are passive tools in which those machines by themselves periodically keep on checking with the monitoring tool and saying … and so on.

All these checks they lead to events and events lead to actions, and actions could be sending an email that something is down/up or something not working in certain nodes, like this space is filling up that's how monitoring tools work. An example of monitoring was somebody pointed out was Nagios and Cacti.

Cacti-2 is a tool from Nagios but it is with the new set-up, so if Nagios is the only tool available for you, you can use Cacti, it works with Nagios without complication and you can do checks like this. You can do active checks, you can check SQB or you can check with NRDE agent which runs on that machine.

You can check the CP on this or you can have some passive checks going on and you can notify or can write some scripts and run them, when something has happened. The good thing is I have one example of a Cacti running over here and I just show it to you and we try to do it quickly because we are running out of time.

This is a Cacti server which we have running and over here if I take you to service details you will see that we are looking Hadoop thing and we are able to see that Hadoop is running, as DSFM, YAML is also running. As DSFM is running it looks up this particular port and for YAML it looks at ATT fold; if I actually take it down, I should take that list of this machine down … Then after a while you will see that it will starting saying: I am unable to ping or do something over there.

It takes a while, it doesn't just out rightly say that things are not running, but after a while it sees that things are not running. We do it a couple of times and we will definitely see something turning, red over here … So instead of the ping it has gone red, slowly everything will just go red and this will report things are red. This is a good tool to start with and both Nagios and Cacti are very old, is a very old concept.

Nagios is really very old and other tools also but like I said that is more on network heavy. So ZenOS, Hyperic; New Relic and many such tools which are upper level. Before we go further I would like to talk about another tool called a Pager Duty. We like this tool a lot because it a build up on all these tools, whether you are using Nagios or ZenOS or the Relic or anything; everything.

What it does is, it's a cloud-based tool which takes input from all the tools and then it's like a pager service to the people where you are. Let's say you may be responsible for keeping the server up and he is located in the U.S. time zone and somebody else is located in the India time zone. If he is not available then better than to try and ping him and wait for a response from him, if he doesn't respond a ticket gets escalated and it goes to the other time zone person, if that doesn't also happen, when somebody at a higher level, it gets escalated.

Page Duty is very good in that particular thing, it has got a mobile service and a lot of other things, and the good thing about Pager Duty is even can configure it with many kinds of services. So if you had to really add a new service, just check what kind of use really from Amazon Cloud Watch to Nagios to Cacti you will see the support for everything over here.

What you do is you add your contacts and then you have your escalation policies and you have your users, and that way you can make sure that you're not only monitoring things right, but you are being updated about everything. It has its own dashboard making everything make sense, which happens, so that is a very good tool you have.

Having said that I'll just quickly go to the next slide then hand control over to Subu. The operation guide is also available that is why they are also being called as DevOps. It is a must so if you want to take baby steps make sure you have automated environment and everything is automated and you make sure that you measure everything. DevOps is progressively elaborate so nobody really has the right kind of answers on day one.

You have to wait and test. The final conclusion is that if you really want to do a faster time to market, more reliable, less stressful recycles, more robust development, develop SAAS you have to start with DevOps. Having said that I will turn over control to Subu, where he can ask questions and he is also going to talk about Synerzip; over to you Subu.

Subu Sankara: Thanks Rohit, thanks for that wonderful presentation. I request the audience to type in their questions and while they are doing that I would like to discuss Synerzip in a nutshell. Again I am Subu Sankara, we operate Synerzip from Dallas, Texas as well as the Bay Area. We particularly are a software development partner for small to medium sized venture backed companies.

We put together a high caliber dedicated team for each client and we reduce the risk of development and delivery by adhering to development technology. We do a little variance of software development depending on the need and the requirements of our clients. Being an Agile model definitely is very cost effective and we guarantee 50% of cost saving advantage by doing the offshore model.

We offer long term flexibility for our customers, our clients [inaudible 0:55:17]… Like if you have a desire to build a team of your own down the road, one or two or three years from today, we can easily build that for

you operate it for a while and then based on the contract, we can cut a deal with … These are some of the flexible offerings from Synerzip and if you … Would go to the next slide …

It is a small representation of our clients over the many years, some are current clients, and some have been acquired by other companies, so this sort of gives a representative list of customers. Again my contact has been provided so feel free to ping me any time, I am Subu Sankara, subu@synerzip.com is my contact, as I said I am located in the bay area.

With that we go back to the webinar topic and one of the questions that has been asked, Rohit for you … How big is the DevOps component of a team logistically?

Rohit Ghatol: DevOps component of a team, roughly I think from my own experience what we did actually, a team size of five to seven, typically we used one person as DevOps basically, a person who work between the operation guys and the development guys, that's what we did in one of our models, typically for a team of five people, one Develop person is a good thing to start with.

As you are going ahead, and ahead the number of people you require in DevOps will decrease than increase basically, because you already developed the infrastructure of the team in your organization.

Subu Sankara: Okay, and the related question is team split into many agile development teams, like four or five or six depending it is initiated for the teams working on … Does each agile team need to have a DevOps person or is it one or two as a common who have the DevOps component?

Rohit Ghatol: Particularly DevOps would be common properties and that reason behind that is a division between products. You never put up one product and I am not in a case where things are really so isolated. They are working either together or they are working on a similar platform or a Cloud or modulization service. So having DevOps team common is a common thing, depending on the life cycle of that product, which is embracing DevOps.

The DevOps team focuses on that product then it moves from one product to another product. Once the initial hurdle is passed, the same DevOps team and then basically help all the team simultaneously and remember don't be shy to do cross pollination. Don't be shy to make DevOps a part of developing or one of the developers part of your

develop team for a while, so do cross pollination whenever required. With that having a common DevOps is a common ground.

Subu Sankara: All right, very good Rohit thanks for the wonderful presentation. We are at just about time to end our webinar, so I really want to thank our attendees for taking time from the busy schedule to attend this webinar. We will definitely keep you posted, we do monthly webinars on technical topic or a business topic related to agile mobile social networking and big data. We hope to see you in one of the future webinars and we really want to thank you in the meanwhile for taking time to attend this webinar.

Rohit Ghatol: Thank you.