

[Very cool! VeeamON Virtual Conference](#)

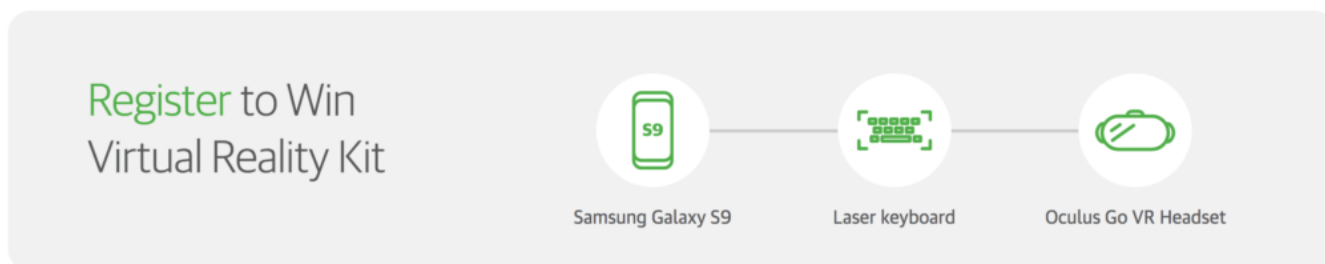
One of the greatest things about conferences is the ability to get very up-close access to engineers and other technology professionals. What makes the [upcoming VeeamON virtual conference](#) on December 5th a double-win is that you get that access but you don't need to book travel and get approvals for budget!

Virtual Event, Real Value

Sign up today for the event and this will also get you access to great live content, post-event recordings and materials, plus there are prizes given away throughout the event day which is always a nice bonus.

There are three tracks (Business, Technical, Cloud) that can satisfy lots of different attendees and audiences. I'm honored to know most of the folks presenting, so I can attest to their skill and presentation abilities. This will be a must-attend IMHO! There are not many times where you can get to interact with people at the event from the comfort of a chair and switch tracks without having to run from one building to another.

Did I mention prizes?! This is pretty awesome...



There's definitely nothing to lose in joining the event. [Register for free at the event site \(https://go.veeam.com/veeamon-virtual\)](https://go.veeam.com/veeamon-virtual) and please do let me know what you thought of the content as I'll be watching it closely myself. Always great to share notes!

[Two of Your Nines Don't Need Five Nines](#)



The re's a pretty good chance that you have a number of environments that are tagged as having a five nines requirement for availability That don't actually need it. We often find ourselves getting confused with availability and the true cost of keeping application environments up on the ever elusive 99.999% uptime.

Breaking Down the Nines

It is eye opening when we break down the actual downtime allowed in the sense of minutes per year when we look at the level of nines as listed on Wikipedia (https://en.wikipedia.org/wiki/High_availability):

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
90% ("one nine")	36.5 days	72 hours	16.8 hours	2.4 hours
95%	18.25 days	36 hours	8.4 hours	1.2 hours
97%	10.96 days	21.6 hours	5.04 hours	43.2 minutes
98%	7.30 days	14.4 hours	3.36 hours	28.8 minutes
99% ("two nines")	3.65 days	7.20 hours	1.68 hours	14.4 minutes
99.5%	1.83 days	3.60 hours	50.4 minutes	7.2 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes	2.88 minutes
99.9% ("three nines")	8.76 hours	43.8 minutes	10.1 minutes	1.44 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes	43.2 seconds
99.99% ("four nines")	52.56 minutes	4.38 minutes	1.01 minutes	8.66 seconds
99.995%	26.28 minutes	2.16 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	5.26 minutes	25.9 seconds	6.05 seconds	864.3 milliseconds
99.9999% ("six nines")	31.5 seconds	2.59 seconds	604.8 milliseconds	86.4 milliseconds
99.99999% ("seven nines")	3.15 seconds	262.97 milliseconds	60.48 milliseconds	8.64 milliseconds
99.999999% ("eight nines")	315.569 milliseconds	26.297 milliseconds	6.048 milliseconds	0.864 milliseconds
99.9999999% ("nine nines")	31.5569 milliseconds	2.6297 milliseconds	0.6048 milliseconds	0.0864 milliseconds

So, when we think about the cost of maintaining an uptime level for the application, it becomes important to see the real numbers in relation to availability. The cost of achieving 99.999 versus 99 is significant.

Fine Nines, Nine to Five

This is the real situation that many of us need to deal with. While we talk about the criticality of an application environment, it's often thought about as critical only when it's in active use. Most folks would obviously look at a straight 99 percent uptime with 87 hours and balk at that as a suggested

availability. Here's the catch, though. What we are really looking for is a five nines availability, but only during access hours. Many, if not most, of our internal business applications are only accessed during the day inside office hours.

Even if we span across time zones, the reality is that we aren't using the applications during a decent amount of time in the day. Assuming that your application needs to cover time zones that span a continent, you are probably needing to cover a 10 hour day with a maximum of a 5 hour variance, totaling 9 hours a day that it is not needed for primary use. That means that you can effectively sustain 2964 hours...yes hours...of downtime. That means 177,840 minutes.

Does this mean we can shut them off? Well, not quite. Let's talk about why.

The Highly Used Unused Application

Applications are considered active during a certain window which I refer to as primary use. There is a non-primary use set of processes which happen as well.

Backing up the environment is a good example of this. Backups tend to run off hours so as not to collide with primary use inside business hours. Security scans and other safety practices also take place outside of core hours to help with keeping application performance more stable during primary use hours.

Because of this we do still have requirements to keep the application platforms available to run these other operational processes.

Scale-back Versus Shut Down

As your application environments are being architected or refactored, it is good to think about the importance of a microservices approach and why it can help with this issue. I know that there are assumptions around the fact that we are choosing when system availability occurs, but the important part of this discussion is that you may be paying for a surprising amount of warranty on systems that don't need it.

We can see that we can't really just power off servers a lot of the time because of the backups, security scans, and other non-primary use access. What we can do is to use a scale-out and scale-back approach.

Web applications may need the back-end to be continuously available but at different levels of usage. During off hours, why not have less front-end servers? Data layers can stay up, but can also be scaled down.

Some applications like file servers and less variable use applications will not do well in scale-up/scale-down scenarios. That's ok. We have to accept that hybrid approaches are needed across all areas of IT.

Why is This Important?

Think ahead. When the architecture is being evaluated for production and disaster recovery, we should be thinking about primary use and availability as well as the non-primary use functions like data protection.

All of a sudden, those buzzwords like microservices and containers with infrastructure as code seem to make some sense. Should you be racing to refactor all of your apps? No. Should you be

continuously evaluating the environment? Yes.

Most importantly, be aware of the true cost of the five nines and whether you really need it for all of your applications.

[Interesting results from a Vision Solutions survey](#)

I'm always watching for stats and numbers in the industry. We are continuously presented with the "fastest growing" and "x% better than the competitor" based on a number of sometimes skewed statistics. While I love what information I can gather from statistics for hardware and software, it is almost always based on sales.

When I was given some statistics recently by my friends at Vision Solutions, I really dug in because these numbers presented some interesting views on what is happening in the technology. Of course, I know that even these numbers may be open to a certain amount of interpretation, but hopefully you can read some of the same information that I have from it.

Consistency across the results

This survey was done using 985 respondents who ranged in company size. Here is the description of the survey participants:



The interesting thing is that we see information which is contrary to much of what is trending in the world of marketing IT solutions. It isn't that all-flash, or public cloud, or any of the specific big trends aren't real. What it does show is that there is a big representation of the "middle class" of technology consumers.

- 51% indicated that storage growth was between 10-30% per year
- Replication features was nearly even at 39% for hardware and 35% for software
- Tape isn't dead - 81% of respondents use tape as part of their data protection solution

There are more details throughout, but these jumped out at me in particular.

Test the plan, don't plan the test

A frightening number that came from the survey was that 83% of respondents had no plan, or were less than 100% confident that their current plan was complete, tested and ready to execute. One word reaction to this: Wow!



As Mike Tyson said: "Everyone has a plan until they get punched in the mouth". Many people that I have spoken to have a long lead up to their BCP/DR test and a significant portion of the planning is

one-time activities to ensure the test goes well, but just to satisfy the test, not necessarily to build a self-healing infrastructure which is where we should be working towards.

To date, one of my most popular blog post series is my BCP/DR Primer:

BCP/DR Primer - An introduction to planning your IT recovery strategy Part [1](#) - [2](#) - [3](#) - [4](#) - [5](#)

This is a clear sign in my mind that people are continually looking for resources and tools to build, or re-evaluate their BCP plan. Since I'm a Double-Take user, the combination of all this hits pretty close to home. I've been using products with online testing capability for a number of years which helps to increase the confidence for me and my team that we are protected in the event of a significant business disruption at the primary data center.

Enterprises love their pets

✘ With years of work in Financial Services and enterprise business environments, I get to see the other side of the "pets versus cattle" debate which is the abundance of pets in a corporate data center. Sometimes I even think the cattle have names too.

Legacy application environments are a reality. Not every existing application has been or will be developed as a fully distributed n-tier application. There are a significant number of current and future applications that are still deployed in the traditional model with co-located servers, single instances, and other architectural challenges that don't allow for the "cloudy" style of failure.

There is nothing wrong with this environment for the people who are mapped to this model today. Most organizations are actively redesigning applications and rethinking their development practices, but the existence of legacy products and servers is a reality for some time to come.

Evolving application and data protection

I'm a fan of Double-Take, so I'm a little biased when I see great content from Vision Solutions ☐ What I take away from this is that there are a lot of us who may not have the ideal plan in place, or may not have an effective plan in place at all for a BCP situation. The content of seeing people's preparation is only half of the story.

✘

Having a plan is one thing, but seeing what the results of real data loss and the reason behind it is particularly important. Using manual processes is definitely a fast track to issues.

Beyond orchestration, the next step I recommend is using CDP (Continuous Data Protection) where possible. My protected content (servers, volumes and folders) are asynchronously replicated, plus I take daily snapshots for full servers and semi-hourly snapshots of file data. This ensures that multiple RPOs (Recovery Point Objectives).

In the event of a data corruption, the corruption would be immediately replicated...by design of the protection tool. Using a previous RPO snapshot prevents the risk of a total data loss by using an automated snapshot. Phew!

Ultimately, the onus is on us to enhance the plan, build the process, and evaluate the tools. If you want to find out more on how I've done data and server protection, please feel free to reach out to

me (eric at discoposse dot com) and if you want to find out more on the Vision Solutions portfolio and Double-Take, you can go right to the source at <http://www.VisionSolutions.com> and there are some great whitepapers and resources there to help out.

[The mighty chain of IT: Where five 9 uptime falls apart](#)

✘ It used to be said that if you want something done right that you have to do it yourself. True words, but unfortunately that only works if you are ready to maintain the entire scope of build, deploy, monitor and support all by yourself.

Earlier this week, GoDaddy.com suffered from an outage which highlighted some significant worries for many people. Whether you were one of the millions of sites hosted by GoDaddy, or one of the millions of customers who use GoDaddy DNS services, you were the unintended victim of a brutal situation.

Regardless of the fact that it was an unexpected attack by a member of the famed Anonymous hacker group, the end result was the same for all of those customers (me included); we realized that the five 9s uptime promise is ultimately on a best effort basis.

Today, prominent programming author and blogger [@JeffHicks](#) was on the recovery from a hacked Delicio.us account resulting in a Twitter blast of spam posts under his profile. While this doesn't affect the uptime of any of Jeff's services and sites, it speaks to the importance of the chain of IT.

Weakest Link

"A chain is only as strong as its weakest link"

We've all heard the phrase, quoted the phrase and seen the true result of it as well. After years of BCP/DR design and implementation I've had more than enough exposure to the SPOF (Single Point of Failure) concept, and the idea that interdependent systems reveal vulnerabilities that have to be understood.

If you were running your application infrastructure and counted on the GoDaddy 99.999% uptime "guarantee" you have now become the SPOF to your customer. It wasn't your fault, nor could you have thought you needed to plan around it really. How much more that a five 9 uptime guarantee could you ask for.

LCD - Lowest Common Denominator

I wrote a series about BCP/DR geared towards the "101" crowd who may not have had exposure to a fully featured BCP program. In one of those posts I talked about how the [Lowest Common Denominator](#) is what you use to define the recoverability and reliability of your service.

As we evaluate our business and application infrastructure we have to understand every component that is involved to fully realize where we have exposure to failure or vulnerability.

Known knowns

Donald Rumsfeld had a great statement about what we know. This is what he said:



It's a powerful statement and I've used it many times in presentations. I've been asked by management teams over and over again (usually immediately after a system failure): "How do we plan for unplanned outages?".

It is ironic that we are trying to plan for something unplanned. The simplicity of the statement almost gives it an innocence. But there is truth to what it asks.

Test Driven Infrastructure

In a previous post about [Test Driven Infrastructure](#) I promoted the use of TDD (Test Driven Development) methodologies for building infrastructure and application systems. It's an important part of how we get to the four 9 or five 9 design. We cannot just throw down a "guarantee" or a "promise" of uptime if we do not fully understand what it means.

The ideal case for any system is that we can design, build, test for failure and then and only then do we really see the potential uptime. If you've been involved in BCP, you also understand that there are levels of failure that we plan for. Somethings are beyond the ability to plan for or are so cost prohibitive that we can't implement the "perfect design".

So what do I tell my customer?

We can only speak to historical uptime. Have you heard the statement "past performance is no guarantee of future returns"? We also generally don't expose our entire end-to-end system design to every customer in every case because it would be challenging, and nearly impossible as systems change over time.

As a provider of services (whatever those may be) you will be committed to some SLA (Service Level Agreement) and as a part of that agreement you will have metrics defined to say where we pass or fail. Another key part of that will be a definition of what we do when we miss an SLA. Do we define our SLA over a week, month, year? It's a great and important question.

What now?

I don't want to sound like a negative Nelly, but I do want to raise the awareness of designers, programmers, admins, architects and management all over that we need to do our best to be aware of vulnerabilities, exposures and this may move into privacy and security which are ultimately part of the overall picture.

Not too long ago, Dropbox suffered an exposure because of the simplest possible thing: employee password hack. Regardless of their globally distributed systems and highly available systems, a single password opened the door to a potentially fatal breach.

So to refer to Donald Rumsfeld, we have “unknown unknowns” that cannot be accounted for, we also have many “known unknowns” that we can get closer to understanding and preparing for.

Break out the Visio diagrams and take a deeper look into where you may have some exposure. And as you do that, you may realize that it is just part of the design and is unavoidable, but it is better to know than to find out the hard way.