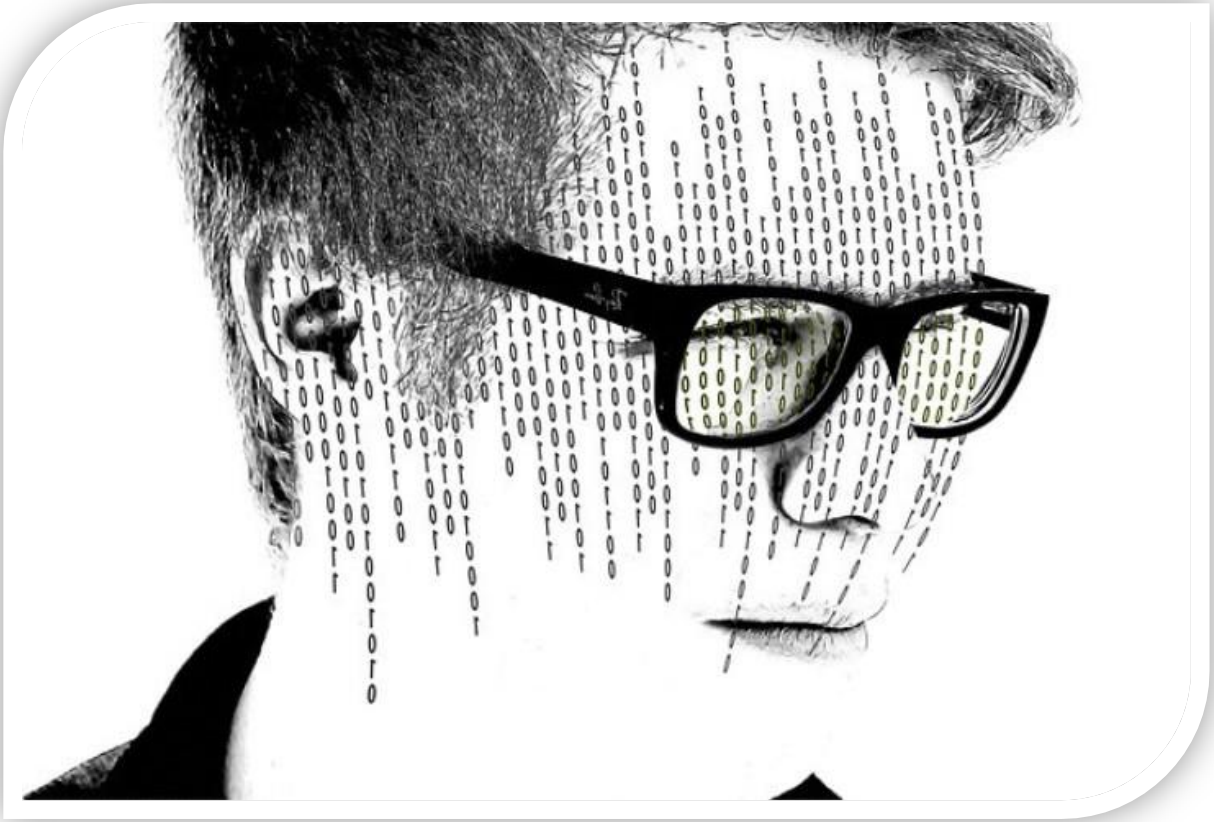


# How-To Guide for Building a DevOps Start-Up Plan

**By Joe Sanchez**

# How-To Guide for Building a DevOps Start-Up Plan

## Table of Contents

## What is DevOps?

*Contrary to popular belief, DevOps is more than automating code deployments and releases! It's the culmination of behaviors, community, culture, and technical talent; colliding to improve IT Services through tools, technologies, trust, and people.*

**Kick Starting Your DevOps Plan...**

## Introduction

It's an understatement to say DevOps is the **"buzz"** right now...

But can DevOps solve your operations and development problems? Maybe.

Books, Blogs, and Podcasts are flooding the Internet, LinkedIn, and Amazon - and the buzz is only going to increase now that the vendors have jumped in.

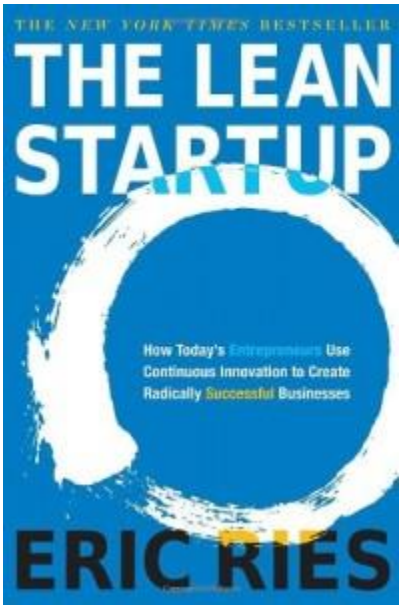
In this DevOps Guide, titled ***Shades of DevOps: Kick Starting Your Plan,*** I'll lay out some ideas and markers for those interested in considering this path.

But before we get started, I want to share something up front about this guide. There's no magic system included for how to get 30,000 releases per hour without crashing your services. If this is what you want I suggest you do a search for continuous delivery, Chef or Puppet.

No, this eBook is more of map from point A to point C, which gets you started. Getting from D to Z will evolve over time based on your needs (and how quickly people adapt to change).

Good Luck on your Journey!

Thanks, Joe




## Shades (Overview)

### What qualifies me to write this DevOps eBook?

Mostly intrigue but also having participated in 4 very different implementations of DevOps since 2009.

I've seen firsthand the good (and not so good) decisions leaders have made, which now in hindsight just goes to show we are all just figuring DevOps out.

Let's see.

Two of my DevOps experiences started very successful, and then went downhill very fast. In my opinion it was widely due to a lack of understanding, egos, grandstanding, and hang-ups.

*BTW, hang-ups are a big part of the problems faced and I will cover them a lot.*

The next experience had partial success but never really transformed into the shade of DevOps everyone is infatuated with (For more about this infatuation, read The Lean Startup).

And the fourth journey is underway and seems to be working - although, there are still hang-ups to be worked through.

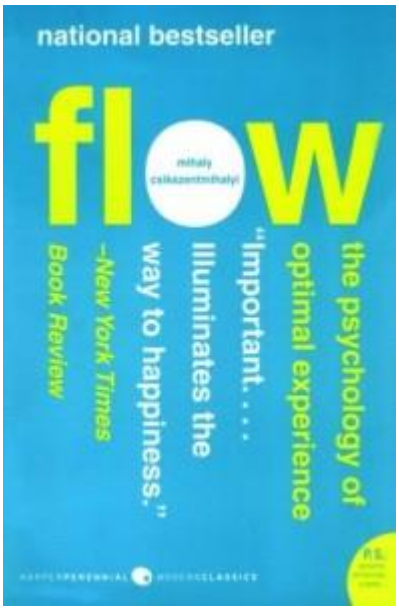
I admit. **I'm no Expert** by any means, but I do have experience.

My suggestion to you is this – "Eat the meat and spit out the bones!"

## Setting Expectations

***In Shades of DevOps: Kick Starting Your Plan***, we'll visit the visible culture challenges and hang-ups I've encountered during my experiences.

*Honestly speaking, my hang-ups were [and are] also part of the problem, which is another reason I can write this.*




We'll discuss, as the titles suggests, how there isn't just one shade of DevOps, there are many.

*A word of advice, don't get hung up on what everyone else is doing or posting on YouTube or blogs [even my words]. Focus on finding what works for your organization.*

As you've probably already found during your research, all some groups want is 1,000s of code releases per hour. This doesn't make any sense to me - but hey, whatever works to solve your IT problems!

Then there are others who want automation so they can leverage staff more efficiently. I find this not be as easy as it sounds, mainly because it takes the same SysAdmins (or head count) to maintain the automation.

And then there are managers like me, who want flow. Flow is a high state of productivity. It happens in individuals and in groups. It's what makes 8 hours feel like 15 minutes. And it gets things done. That's what I want, getting things done.

I'm sure there are many more reasons but like I said whatever works to solve your problems.

*Here's a tip. Be willing to experiment. And fail. Then try again.*

I know this sounds ambiguous, right? It's because DevOps isn't your typical framework.

As many have already found while searching the web for plans and guides, changing a legacy IT Department “way of doing things" isn't easy. And in some cases, it may not be worth the trouble.

But for groups willing to press-through the DevOps startup phase, the value-add is a high performing organization, new culture, and a better service delivery system; which translates into happy customers.

And happy customers are really what we want, right?

My goal with Shades of DevOps is to help open your mind to things already happening around you and to a new way of approaching IT.

## White, Grey, Black

Let's focus for a moment on the title of this eBook **“Shades of DevOps.”**

The title perfectly sums up DevOps because there isn’t a gold template way of approaching it, anyways not yet.

In this eBook I'll do my best to make things easy to understand and beneficial for my readers.

Let's start off with explaining (3) basic approaches (shades) you can take.

- White - Dipping your toe into the murky waters but you’re not really ready to get wet. Now you can actually say you’re using DevOps but you really aren't. This one is more of a prop.
- Grey - Wading in slowly but only going waste deep. No real value has been gained except finding the threshold where the pain is too much to bare. Many have gone here and quit.
- Black - Jumping in head first and creating chaos. This shade results in clashes between Ops, Engineering, and Development tribes. “OMG, what have we done?" Roll back!

All joking aside, what works for one organization may not work for another. Why? It’s because DevOps is more of a culture than a methodology, software tool, or even a practice. And every business has different people chemistry.

Some will adjust, while others will quit and leave. Not everyone will want to be a part of the change. Heck, ***you*** may not want to be part of it either but someone above you said “do-it!”

**Starting Point**

Irrespective of your business, culture, or people, the point on the map where we can start from is - ***You***.

Start by removing your Ray Bans so you can see clear!

Open your mind and really consider what I'm about to say.

Ready?

***IT Operations [and Engineering] see Development as a "THEY" instead of an "US." We are the problem.***

Am I right?

They're blaming us, we're blaming them. And nobody is happy, including the customer.

We'll go deeper into this point later on but for now let's dive right in and address some hang-ups we need to get past.

The first elephant in the room - **CONTROL**.

## Hang-ups (Obstacles)

### Are You A Control Freak?

Keep this on the down-low, but I've been in closed-door meetings with executives and I've heard the excuses.

Process, People, Resources, Time, Cost, etc…

How do we maintain control of our staff and resources when the solution to solve the service problem is a sharing culture that collaborates extremely close with [developers]?

Obviously, many will disagree with my views here [especially vendors], but that's okay.

And here's why they might disagree. It's because they want to sell/buy IT solutions (shiny new objects).

Buy this software tool, hardware gadget, professional service, or training and your problems will be fixed…

Absurd! The truth is buying stuff helps IT maintain control.

Now we can say we're working on the problem. But the problem between Dev and Ops keeps growing.

Sorry to rain on your parade but the real solution for a service delivery problem cannot be purchased.

I will even go as far as to say it will be impossible for some organizations to transform because the people trying to institute DevOps will not relinquish "Absolute" control. You can quote me on that! I've seen it.

*Let me warn you. I tend to get sarcastic but only because I am passionate about what I do for a living. If I whined you up, stop and take a break. Get some fresh air. But then keep reading because we're just warming up...*

**Kryptonite Drains Opportunity**

Let me continue where I left off, talking about control. Here's what I am saying. Control is a hang-up. And in DevOps it's like Kryptonite in a good Superman story. It's a hang-up that drains opportunities from DevOps.

I've seen it happen before. One day someone high-up decides they are “now a DevOps shop" and mandates you form an exclusive team, their team.

So you set out and cherry pick all the best talent, call them “Team DevOps” and get them t-shirts, right?

Next you remove all processes and turn them loose. Team DevOps has immediate success but then something unexpected happens. Chaos and resentment ensue (remember the shade of black I spoke of earlier?).

Why do you think this happens?

I'll tell you what I think. It’s because people are people. Eventually jealousy sets in and the un-involved start blocking progress, while the control freaks find new ways to regain control.

No, DevOps is more of a decision process you start making months in advance. It starts by changing how you think and approach IT.

Control is only one hang-up that will stop progress, there are more. Coming up next in the retrospective I'll provide a list of hang-ups for you to consider.

***Congratulations, you survived so far! Now, I'm going to raise the bar and get personal.***

## Retrospective (Hang-ups)

You've read this far so I'm assuming I can start getting slightly more personal. Heck, we're almost drinking buddies now (except I don't drink).

Take a deep breath because we're about to look at other hang-ups. Some may be painful [or even insulting].

1. How do you treat developers? Why haven't you scheduled regular meetings with them so you can learn about their pain and frustration? Here's why. Because most of the time we assume to know what our customers need, wrong! Creating a feedback loop is important to your success.
2. Are you protecting any Prima-Donna's who nobody wants to work with? Stop! And get real with them because they are roadblocks to progress.
3. Are you focusing on what is important based on delivering great services to your customers, or are you building "The Great Pyramids?" Pyramids are great for egos. They make you look like you're accomplishing great things. But in reality they add very little value to the company. (Ah, but you look good!) Keep reading for more details.
4. Who benefits from the projects in the queue or backlog? It's time to re-prioritize and focus on what the customer needs. Listen to the feedback and stop wasting your team's time on projects that add little or no value.
5. Do you have any of your most talented people meeting and working with your customers daily so osmosis is happening? My point here is seeing and feeling the problems firsthand will help solve them. Get your stars into the trenches and off of your pet projects...
6. Are outside forces such as vendors or political agendas blurring your focus and causing you to make "BAD" decisions? BTW, an MBA doesn't make you immune to foolishness!
7. Are your leaders, managers, and executives part of the problem? Are they only telling you what you want to hear? [Read the emperor's new clothes] Is this you?
8. Do your staffing processes weed out people who are, or will be, counterproductive to a DevOps culture? I'll cover this deeper when we get to DevOps Interview Questions.
9. Are you rewarding players who collaborate and are cultivating teamwork? Bonuses are great incentives, but so are compliments!


**10. Finally, ask yourself this. Are YOU the problem?** If you're a Prima-Donna with a political agenda and you have everyone buzzing around you building the Great Pyramids then forget about DevOps solving problems because it will fail! I only have one thing to say to you, deal with your hang-ups!

I warned you I was going to get personal. Hopefully you're taking my comments constructively? If not, leave me a comment.

**Be real about your hang-ups.**

As we come to the end of this retrospective, I'm not asking for you to agree with me because I know in the end - you will do what you always do. I've seen it before and it won't surprise me if it happens again.

Seriously, why do you want DevOps? If your goal is only faster and more releases, you're missing the bigger picture. It's about **You**!

### DevOps References:

- What is DevOps? Infrastructure as Code by Mike Loukides (Download Free eBook)
- Adrian Cockcroft's article ( Read: NoOps at Netflix)
- John Allspaw's article (Read: Rebuttal)
- The Phoenix Project by Gene Kim

### TRUST...

Trust [or lack thereof] is the next topic we'll look at.

Prepare yourself because now we're going to go deeper into the grey areas holding IT organizations back.

***The absence of fear is trust.*** Let that sink in…

Let’s face reality. We are scare-to-death of trusting, which is why we become loaded down by rules and processes.

This is why ITIL is a dream-come-true for control freaks.

We’ve heard the tales (or read the books and blogs) dramatizing failed ITIL implementations. Most of these stories are probably true. But they are no reason for not changing.

The key for any successful service improvement plan is balance.

What's feeding your fear are thoughts of losing your precious control.

Yes, I'm talking about control again.

The solution for this hang-up is TRUST.

*Here’s my point. Trust conquers fear and brings the balance we need between control and process.*

Sorry if I'm being too philosophical, I was having a Zen moment.

**How trusting is your organization?**

## Retrospective (Trust)

How trusting should you be?

The answer varies, but a rule of thumb is as trusting as possible while not putting your services at risk.

The answer also depends on the underlining infrastructure and the quality of the code in place.

If the environment or service is fragile (meaning it crashes for little or no good reason), obviously more control is required.

On the other hand, if it is stable, then more empowerment can be allowed.

The bottom line is processes should fit the use case; otherwise they are just additional weight to be carried.

As you reflect on the problems you face, you should start to see patterns similar to those I've covered where people or processes are too controlling and limit, or even frustrate the people doing the work.

You may even be surprised how un-trusting your culture is...

**This is a very important shade to consider while designing your plan.**

## Can you hear me now?

Remember the Verizon commercial with the guy who would go around to different locations and say, “Can you hear me now?"

That's called asking for feedback and it helps figure out what is not working.

Here's a rule of thumb. Listen to **feedback** whenever you get it from the people doing the work and from the customer. Chances are there's something going on you might want to know about.

Even if the feedback is painful - listen to it.

Honestly, even ITIL works if you do these 3 things:

- Understand less is more
- Balance the need for control with trust
- Listen to feedback from customers and staff

Here's a video of Gene Kim. He's become a DevOps icon for his book, The Phoenix Project.

Take a break and watch it all the way. He'll cover the feedback loop and other important items we'll go into later.

**Video URL to watch:** https://youtu.be/9jD200ZxIrQ

That was great video, wasn't it? It really helped me with my thinking.

**Let's move on to the retrospective.**

## Retrospective (Feedback)

Start by looking for opportunities to trust your systems and people!

Do proof of concepts and pilots to determine the level of control required to release new services. And make a list of what people are saying. You don't have to do everything they say but listen.

Never implement processes only for the sake of it. Or because someone with a title says it is needed. Try and reason with them with facts and feedback.

Better services delivery is achieved by reducing complexity. Remember having services that do not need heavy controls and processes is the goal. This happens when we listening to feedback.

***Add this point to your plan. Apply common sense and focus on making things run simple (less complex).***

Next we're going to look at a couple of technical hang-ups many organizations struggle with.

### Uptime!

Which is more important: **Uptime** or **Downtime**?

This isn't a trick question. Which is more important?

*Here's a hint. It's easy to get hung up on the numbers game and start focusing on 9's instead of what's really important.*

The real measurement is calculated in the impact to the customer. What about the people who can't log into their application, or can't work because the app is so darn slow they are ready to lose their mind?

My point it this. What's on the other side of uptime (99.999) is downtime or (.001), which translates into lost minutes, hours, days, productivity, and dollars. Not to mention angry customers.

**Downtime Equals the Impact to Users!**

You know something? I don't recall any Uptime classes as criteria for my Computer Information Systems degree.

Ironically, thousands of technology students are graduating every year and entering the IT world without knowing what the most important objective of their job is!

**Let's try it again.**

What is your most important objective?

*The server, No! The network, No! The code, No! It's keeping the user or customer happy! The person paying the bill is why we do what we do.*

**Let's cut to the chase.**

I'm not going to tell you how to reduce downtime, but I am going to tell why you need to understand why it's so important for your company to improve uptime so your valuable customers are not having downtime.

***Add this to you startup plan because this is another reason organizations need DevOps!***

## Retrospective (Uptime)

Server vs Service; which one is more important?

For most VMware, Windows, or Linux admins, this is a no brainer – the server is the most important thing in their daily lives.

They take pride in designing and building servers and then give them cool names like Ironman and Thor, right?

Do you see the hang-up with this way of thinking? I'm not saying not to take pride in technical accomplishment but obviously this mindset cares more about technology than customers and this is what fuels uptime reports.

Shifting the mindset to the customer side and feeling the pain customer's deal with will have incredible results on improving uptime. Try it.

Caring is common sense, but not common.

**Next Up...Stop throwing tickets over the fence!**

## Easy Button

Long before there were frustrated developers waiting days or weeks for VMs, there were frustrated developers waiting days or weeks for RAW metal servers.

So what has really changed?

Or here's a better question. What's still wrong?

Wasn't virtualization supposed to fix the server delivery problem?

Yeah, kind of - but the server build process is still hitting roadblocks because although virtualization improves hardware utilization it doesn't fix the build steps IT groups create and jump through to deploy a server. This is another hang-up for the list.

Most developers don't care what hoops you jump through, or whose label is on the hardware, or which virtualization software you're using. If you can't give them the compute power they need when they need it you are failing.

**What would Google do?**

Both Amazon and Google have been building their own (non-branded) low-cost and highly redundant infrastructure for years.

They get it! Their focus is on making their services as easy as possible for customers to use. Google doesn't boast 3rd party brand names, they boast simple.

And while Google is making things simple, the traditional IT department who has purchased every whistle and bell Cisco, NetApp, VMware and other vendors offer is still struggling with a server deployment workflow that manually transfer tickets around to do tasks.

Haven't we learned this lesson, yet? Easy works better. These gaps are causing Dev Teams to give up on Ops and shift full speed ahead to deployments in the cloud, public cloud that is.

***Put this on your list. If you have issues with server deployment it needs fixing ASAP.***

## The Typical Server Build Workflow

Let's drill into what's going on behind the scene when servers are requested.

Here's a list of common steps followed for building a "vanilla" VM server.

**Day 1:** It starts with a ticket getting created (and God forbid if this doesn't happen first!), then meetings, then filling out of forms, then waiting and following up. And then more waiting and even more follow-up, etc.

- Carve out the VM, CPU, Memory, NIC and disk space (first step in the process, granted the capacity is already available)
- RDM, iSCSI or NFS mounted storage is created (done by storage team on a separate task or ticket)
- IP Address and DNS reservation is created (done by the network team on a separate task or ticket)
- VIPs and Alias are created  (yup, done by another team or ticket)
- App, Web or database installation and configuration (yup, done by more teams and ticket shuffling)
- Security scan (yup, done by another team and more ticket shuffling)
- Monitoring, antivirus and other agents enabled (done by yet other teams on multiple tasks or tickets)
- Release to Production Check or QA (a task to make sure everyone has finished their tasks and tickets) Note: This doesn't include installing any applications.

And the list can go on and on with slight differences depending on “controls" and type of server, i.e. App, DB, Web.

**Day 24-30: Server goes live!**

Do you see what's wrong with this picture?

We virtualized the hardware but we left all the legacy hang-ups in place.

So instead of making the server request process faster we really haven't cut much off the provision time except for the racking, cabling and powering of hardware.

Sure the end result is you have built an awesome private cloud but nobody wants to use it because they don't want to wait weeks while each team throws tickets back and forth over the fence to get tasks done.

This alone has led many software development teams to give up on their internal IT departments and go straight to the cloud via AWS, Azure, or Google. There - with a few clicks - within minutes they are coding.

Now granted, there isn't any governance around this yet - and yes, in the long-term it will need to be governed, but for now it's solving the problem of waiting on Ops.

### Let's take apart this problem.

Remember one of the shades we covered earlier?

Here's a reminder. CONTROL.

As we can see in the workflow I laid out earlier, unless you have converged your teams, there are at least 5 or 6 different groups owning tasks in the server build process. And I can bet each does not want to give up the control needed to create the easy button (or at least to streamline the build process down to one team).

What would Google, AWS and Azure do? They would create an easy button. Put in your credit card info, click, click, and get coding!

**Let’s look at a few options we have.**

## Easy Buttons

Here are 3 solutions for improving slow server deployments that are DevOps friendly.

**#1 (Self Service Portals)**

I'm talking about something like vCloud Director or Openstack.

Install the tool, setup the resources on the back-end, and give the power to one team to push the easy button. Portals use APIs and web services to get to the resources needed to create VMs. No tickets getting passed around.

**Gotchas:** The main risk is keeping up with the server sprawl portals create, and possibly running out of capacity.

**#2 - Temporary Converged Team (War Room)**

Whether the resources are dedicated or only grouped together for set times every day or week, they need to be available to complete the tasks together. No ticket waiting days or weeks in the queue. Get it done now!

This keeps the ticket from waiting in the queue for the next task to be completed. Basically it's an assembly line and the end result is servers are turned over to the developers in much shorter times than if tickets are thrown over the fence.

Get all the needed people in a room together. No phone calls, or emails, or IM - they are communicating face to face.

**Gotchas:** The gotchas here are some people cannot be pulled away from the day-to-day for long and too much time cooped up with others can [will] cause conflicts.

**#3 - Automation (DevOps Tools)**

Careful now! Because this group has all the keys and controls from start to finish to push environments out. We're talking server builds, configuration, and deployment all wrapped into one package.

They use technology such as Chef, Puppet, and other automation tools to rapidly get servers stood up. And the same group then works closely with developers to support any config changes and code releases.

The real proof of DevOps can be measured in improved service delivery. For example - shorter delivery times, fewer tickets, closer collaboration, and automated builds, deployment, and release processes.

So regardless how you choose to streamline your server deployment process, the real problem once again comes down to CONTROL.

**Ponder this point.**

While you hold your control, chances are the train has left the station and you'll be playing catch-up with the Dev team already pushing the easy button in the cloud without you even knowing it...

**Add this to the plan. STOP throwing tickets over the fence!**

## Retrospective (Server Builds)

So far we've covered a couple of obstacles faced in by a typical IT organization. And we touched on the effects they are causing.

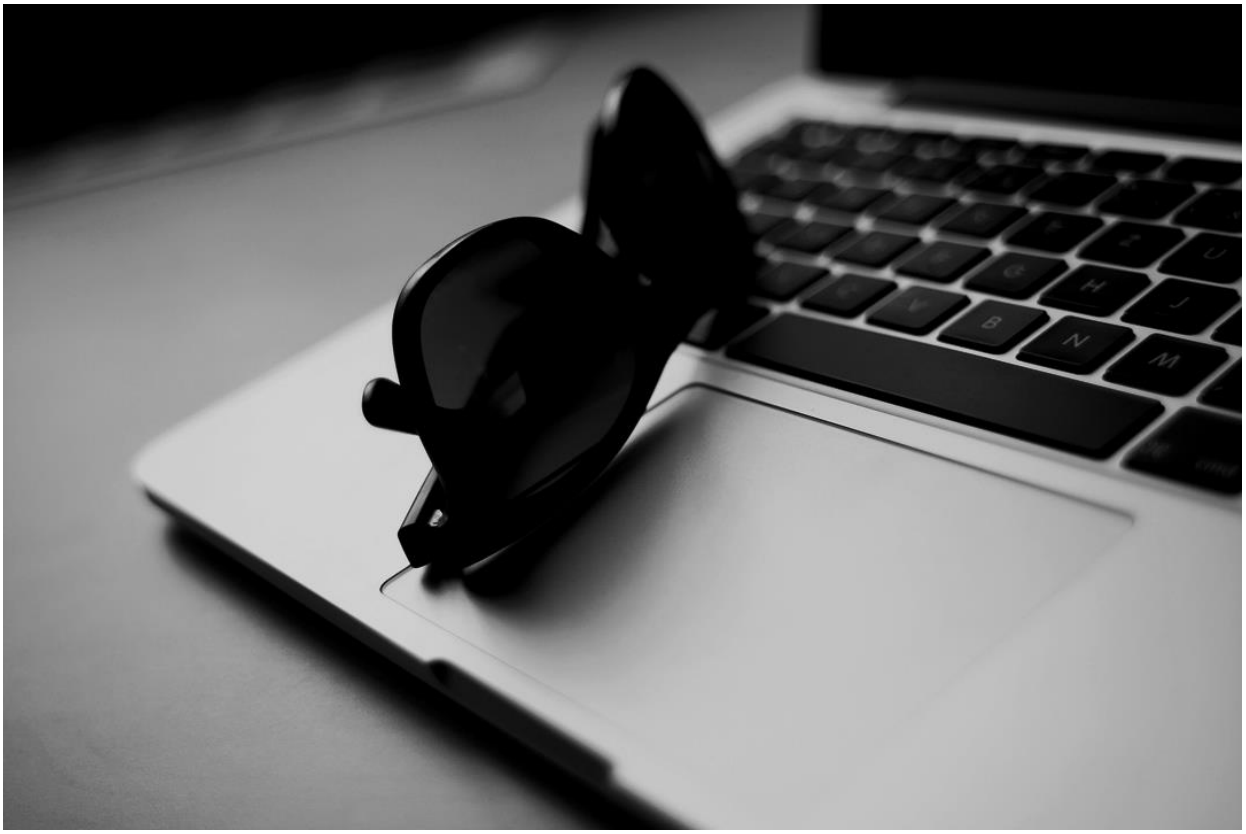
By now the point of this eBook is hopefully starting to resonate and make sense. Either that or you're becoming very defensive, which is another IT hang-up...

Also, by now your minds-eye is beginning to see the various shades of DevOps I referred to in chapter one, and as I predicted, you're thinking you should just leave things as they are, right?

**Here's the good news.**

From here on I'll dive more into the steps you can take for team building (training), hiring the right talent, and for formulating a DevOps startup plan. Now we're getting to what brought you here in the first place, the DevOps plan.

## DevOps Start-up Plan (Beginning)



**Let's see if these 3 examples ring any bells:**

- Months to deploy virtual servers (tickets are being thrown over the fence)
- Failure after failure releasing code (Dev and QA environments don't match Prod)
- The majority of your team's time is spent firefighting!

If you answered yes to all 3 examples, a DevOps transformation could definitely improve your IT service delivery.

Still unsure? Let’s keep going anyways...

## 5 Step DevOps Transition Plan for Beginners

This DevOps plan is concise and only has five steps that will take you from A to C -- not A to Z.

Each of the five steps builds upon the next and there's no time limit involved because it could take weeks, or months, for the transformation to happen. The reason there isn't a time limit is because each "pre-existing culture" will adapt differently.

In my experience it takes at least 90 days for the first sign of the shift to show up. Time is wasting, so let's get started...

### Step #1 - Start with a Culture Shift

Over the decades IT cultures have not changed much and many are still following the same practices that were put into place in the 80s - 90s.

These cultures are normally referred to as "Legacy IT" and though they were very effective years ago, today they lack the agility and flexibility to adapt to fast pace business changes driven by a global economy.

The first step towards DevOps begins with slow-paced group meetings focused on educating departments on the value of DevOps and discussing the current shortcoming of legacy IT. Be prepared because they will defend their turf!

The goal here is to build advocates and enlist facilitators who will ban together. During this time it will be OK to share objections to the changes being offered. Be willing to listen.

After multiple group meetings, lunch and learns, and videos about DevOps; the culture will begin to shift and more people will share the vision.

Also during this time it will be very important to get managers and leaders moving in the same direction. Here is where you will find the biggest hang-ups because these stakeholders have the most control to lose.

This phase of transition is critical and can take months. Don't rush it.

### Step #2 - Create Feedback Loops

What I've found to be true about legacy IT departments is they are extremely defensive.

Ask them why they do what they do and their response is focused more on defending their technologies and practices rather than objectively looking inward to see if anything can be done to improve.

I get why they are defensive. Like I said, I'm in the trenches, too. But it's not working anymore.

Normally there's a history of failures and problems from the past that still cause pain and more importantly, a lack of trust.

You [must] be willing to move past all this STUFF in order to bring about the new alliances and the confidence needed to transform.

Each group or department should slowly begin meeting with their customers (both external and internal) and listening to the pains and grievances. It's not easy, but it will reveal the areas where healing and mending must happen.

In extreme cases some business units may not want your help anymore because they have found another way to solve their issues. Be ready for this and use common sense.

Once feedback does begins coming in it should be logged, and a backlog of problems should be created that can be worked on (I like to use kamban). We'll call this list technical debt.

Another good practice is to close the loop with the customer on any fixes made. Never assume anything is fixed without getting confirmation from the customer. Then once there is closure with a problem, it should be celebrated by the team and marked done on the technical debt list.

No loop should ever be left open. Step 2 will continue as long as it takes and will eventually become a fundamental part of maintaining the DevOps way.

This stage can begin in the first 90 days and the key here is to ***stop being defensive.***

*This raises a good point. Never let anyone on the team ague with the people you are trying to reconcile with.*

I get it, it's not easy listening to someone call your baby ugly but you need to listen to what they have to say even if you don't believe it's true.

The goal of this DevOps startup step is to learn to listen openly to customer feedback for opportunities to improve. Open the channels so they come to you instead of going around you and straight to a VP or the CIO.

### Step #3 - Get Things Done

Another hang-up with legacy IT, “silos".

What are silos? They are technology boundaries that shouldn't be crossed by other technology teams. Kind of like TURF. Networking, server, storage, and data center teams are silos.

Problems arise when silos work on (and building) their own processes without engagement or consideration of each other.

This is why something as simple as building out a virtual server can take many weeks due to the rigor of requirements created by a silo.

*Note: This all happens at a cost to the business and developers who sit waiting on Ops to get it done for them.*

If anything, this is one of the most important value adds that comes from a DevOps transformation. And yes, unfortunately, there will be some silos that will need to be torn down before this phase is over.

Some groups call this collapsing of silos - convergence, whereas I call it DevOps.

The goal here is getting people working together for the greater good. And to work down technical debt (backlog from step #2).

Tip: A highly effective way to make immediate improvement is to use embedded resources. For example, dedicating a person to sit with the development team (or other technology team) and work directly with them. This helps build closer relationships while at the same time exposing them to the pain caused by silos.

Other practices I currently use are: Kanban board for tracking tasks and having daily stand-up meetings. This helps pull people together and address issues quickly.

### Step #4 - Easy Button

What would DevOps be without improving the turnaround time for releasing code?

As many have found as they researched DevOps, a key value add is the automating of software and code deployments and releases. This normally includes using special tools to build environments fast from templates or scripts.

Many of these tools and skills are new, and are in high demand for businesses moving to public cloud. Once the culture and feedback loops are in place, improving processes can begin.

The goal of this phase is finding talented people who have expertise with tools such as: Chef, Puppet, Vagrant, Github and other similar automation software.

You'll need people who can script, and code for REST, SOAP, and other web services which are required to build automation.

Hiring new staff or training may be required to fill this gap if you don't already have the right skills on the team to create the easy button(s).

### Step #5 - Continuous Improvement

Once your IT department begins the journey towards DevOps, be prepared because there will be push-back, frustration, resignations, and disruption. This all happens because you decided to change the status-quo.

Many will find the changes too painful and will give up for the easier path of legacy IT. And in some cases - maybe there is another solution for them, like ITIL.

However, for those who wish to really test the boundaries of new technologies (The Cloud), tools, and processes; continuous improvements can start happening as more and more daily systems are transformed and become stable and efficient.

Each improvement (investment towards technical debt) will start giving back cycles of time which can be used to focus on solving other problems.

At this point you have moved beyond transition and into transformation.

***Warning*** *- Let me share a friendly word of advice. Avoid the temptation to build an exclusive team of players in the beginning who are cherry picked to form an elite DevOps team. In my experience where this was done, it generated a culture of resentment and elitism. The goal of DevOps should be to enlist everyone so anyone can participate and share in the fruits of success. Tread carefully here so you don't make the same mistake.*

**Listen Closely**

I know this DevOps for beginner's guide only scratches the surface. And honestly, DevOps isn't a one shoe fits all solution. It will take some trial and error before you get it going...

But as you begin to map out your transition plan use these 5 steps to help guide you.

You will most likely find there are other steps or actions more important than the ones listed above. This is a good thing because it means you are sensitive to the NEED rather than just following a set of rules or instructions (this means you get it).

Adapting to the need is a true indication you are learning and being transformed.

Continuous improvement is the goal of this step. This means constantly adapting to changes and improving how your IT operations are run.

## Retrospective (Plan)

There you have it. Five steps or phases to kick off your DevOps start-up. I know they don't go into details. But they are ideas not mandates.

Look at it like this.

1. Culture shift
2. Feedback loop
3. Pay-off Technical Debt
4. Build Easy Buttons
5. Keep Improving

It's enough to start your plan. And the good thing is it doesn't need to be complicated. Keep it simple and fun. Don't make it hard for people to adapt to.

My way is to create a “Flow” of processes that belong, rather than making them fit in.

Before I end, I want to provide more information on the skills to cultivate.

**We’ll cover this next.**

# Staffing (People)

## The Elusive DevOps Engineer

Since 2007, like many, I've been involved in IT projects to build private clouds - and other projects for migrating to public clouds.

The common item that continues to stands out from each of these experiences is this:

A new breed of IT person is needed if you plan to advance your business beyond the typical IaaS (infrastructure as a Service) model.

In this chapter, I will share what I have discovered through my trials, and tribulations, over the last 7 years.

Stick with me and I will reveal the secret recipe of skills to help you identify the elusive DevOps Engineer who can successfully get your applications running “RIGHT" in the private or public cloud.

I will also warn you up front, it's going to take more than knowing VMware vSphere and Linux to even get you out of the gate...

## DevOps Job Description

Now before we get to the meat of this let's figure out what a DevOps job description should look like.

First, begin with a DevOps Engineer job search. Go ahead, click the link and check them out.

Do you see what I see? They're all similar except for a few with some unique scripting skill requirements. I plan to ***give you the secret sauce*** recipe if you stay with me until the end. Here we go...

## 10 DevOps Skills To Look for in Job Applicants

**#1 - An Impeccable SysAdmin**

Must be a senior level Windows/Linux Administrator (Either/Or/Both depending on your shop) with 5 - 10 years of experience. Why? Because they need to be able to build and administer servers in their sleep. But that's not the only reason; a lot is riding on someone to automate server deployments because this is a big problem in most IT shops. Remember the easy button?

**#2 - Virtualization Experience**

Must have 3 - 5 years of virtualization experience with VMware, KVM, Xen, Hyper-V, or whichever favor hypervisor you are running in your private cloud.

Now, they may never get involved in the day-to-day support of the infrastructure work, but they darn well better understand it because most public clouds are running multiple flavors of virtualization.

**#3 - Broad Technical Background**

Along with virtualization experience, they must understand storage and networking. Why? Because gone are the days when network and storage are silos. You need people who can design a solution that scales and performs with high availability and uptime.

Applicants also need to understand fault tolerance and failure domains so they are not putting all the eggs in one basket.

**#4 - Scripting Guru**

Have I said they need to be able to script yet? Bash, Powershell, Perl, Ruby, JavaScript, Python - you name it.

They must be able to write code to automated repeatable processes. But we're not stopping there because they also need to be able to code to RESTFUL APIs. That's right, if you are going

to replace manual processes such as assigning IP addresses and DNS reservation, someone needs to write some code.

**#5 - Borderline Developer (more is better)**

Have I said they need to code in C+, C++, .NET, ASP? No, I am not repeating myself. I am talking about writing scripts that will fire off and orchestrate the complete deployments of DEV, QA and Production environments via tools such as Chef, Puppet, CFEngine or other tools of this kind. Why? Because gone are the days when someone installs Windows, Linux or apps from a CD.

Nowadays, you fire off a command that shoots out a server build, which then triggers another script that installs the application, then finishes by licking its lips and shooting off yet other scripts that do configurations and validation checks.

Who do you think is going to write all this code? Not a SysAdmin. But the DevOps Engineers will.

Some would argue he/she doesn't exist but I disagree. The DevOps Engineer is a new emerging role you soon won't be able to be without.

**#6 - Chef, Puppet or other Automation Tool Experience**

I think I already mentioned automation tools such as Chef, but there are others such as Ansible, Fabric, Vagrant and GIT that all have their place on the DevOps toolchain, too.

Finding a DevOps Engineer with all this talent will not be easy or cheap. But let's keep going while I have your attention.

**#7 - People Skills**

There used to be a free pass for people who were geniuses but they just couldn't get along with anyone.

Call them a JERK or other four-letter words, but they were tolerated because nobody else could do what they did.

Not the case in today's world. Fault tolerance and scalability happens at the people level too. And you need people others can go to for assistance without someone taking off their head with insults. We covered this earlier in the list of hang-ups, item 2.

Do your best to find people who can communicate without yelling or fighting. Also, think hard about the people already on the team.

This topic segues into the next DevOps skill which also relates to communication...

**#8 - Customer Service**

If you watched the Gene Kim video I posted at the beginning, then you've heard how important the feedback loop is?

Finding people with half the technical skills I have listed in this chapter will be hard enough, but now I am adding customer service to the list.

Look for people who care and can drill into a conversation with a developer or customer. This is absolutely key to solving problems.

It takes a special person to listen to feedback, especially when the developer or customer is calling someone's baby ugly. The good thing about hiring new people is they don't have the emotional attachment to the current infrastructure or systems. That means they will be less defensive.

Man, I wish I had a dollar for every time a developer blamed my infrastructure for why they were late on a project, or why their app was slow. I think I may have even written a book on this topic already. Anyways, the bottom line is you want people with good customer service skills.

**#9 - “Real" Cloud Experience**

We're almost there. The ninth DevOps skill you want is actual experience deploying applications in AWS, Google, or Azure's Cloud.

Look for real experiences that can be quantified. Finding anyone with any measure of successes is important because there's a shortage on people who understand IaaS versus PaaS; state-full versus stateless, and something known as loosely coupled apps.

It's no longer about fork-lifting existing servers and applications to the cloud, now it's about designing and deploying applications using the “best practices" for Amazon, Azure and Google.

We're talking about doing what the people building clouds are doing, which means leveraging software defined data centers to code true PaaS environments.

We're also talking about putting compute, networking, and storage APIs at developer's finger tips.

**#10 - Someone Who Cares**

I know I'm asking for a lot.

We've come to the final skill which BTW is dear to my heart, and I want to say it's very rare.

When I say someone who cares, I mean really cares. IT people care but we want to be left alone in a dark corner.

Finding someone with this skill is rare and worth every dollar if they can code, too.

**Let me explain what I mean by care.**

- I am talking about someone who can mentor others.
- Someone who is willing to share their ideas and scripts with the team.
- Someone who can lead people and get people thinking together about solving problems.
- Someone who can communicate without becoming defensive.

Far too often the real problem with IT is because people don't talk, or should I say, “They don't listen!”

God gave us 2 ears and one mouth so we could listen twice as much as we talk!

**Now you have the recipe!**

These are the 10 DevOps skills to look for in applicants when you screen resumes and people for the elusive DevOps engineer position.

I can guarantee you; it won't be easy to find good applicants. And you will most likely need a strategy to create the right set of DevOps interview questions, too.

Interview questions will be covered in the closing chapter because there aren't many hiring managers or recruiters around who understand or have the right mindset to write them. And up until recently, even less who understand what DevOps is.

## DevOps Salary

Finally, I want to cover what a DevOps Engineer salary may be. Let's look at a salary graph from SimplyHired.

Seems like the top end of the range might be a little low, while on the bottom end the range looks good.


DevOps Salary Graph Compliments of SimplyHired

## Retrospective (People)

Think hard about what we just covered.

We're not talking about a network or server engineer who might make anywhere from 50 - 80K. We are talking about an elusive skill set not many people in the world currently have.

Finding someone with 6 out of 10 of the skills listed would be a prize!

**Secret Sauce**

How important is it to you to do things right the first time? Or should I say, how much are you willing to pay to do it a second or third time, or until someone gets it right? You know where I'm going.

You see, the hard lesson I have learned in the last 7 years is businesses can always afford to pay twice, yet they never understand the value of paying enough for the right people to do it right once.

Championships are won by the right people and leaders who leave it all on the field, or court, when it counts most.

Here's a rule of thumb. A DevOps salary is more than enough but less than having to pay twice, or three times the amount to do the same work over...

## Changing Your Mindset (Be Unselfish)

As we come to the end, I want to leave you with one final thought.

***Shades of DevOps*** *is really about building an unselfish culture. Here's what I have learned. You will either have success or failure in the end because of people and their mindsets.*

Let's look at a few DevOps interview questions I've written which will help give you a reference point for where we are going in this final chapter.

With each question there is included an explanation that can help get you into the mindset I'm referring to. Start by asking yourself these questions.

One more thing before we go on, let's understand something. DevOps Engineers and Admins are not really job titles…their ideas. I know I said earlier it is a title but for now they are ideas.

And, if you've spent any time listening, DevOps is not a team; it's a culture and/or philosophy for improving IT service delivery.

I'm going to give you another good resource for this DevOps, check out the DevOps Cafe Podcast (Hosted by: Damon Edwards & John Willis).

Want DevOps history? Go to their archives and listen to some of their first shows. I love these guys!

What you'll discover is DevOps is about synergy and the goal is collaboration, teamwork, and flexibility.

**Now you're ready, so let's get to the questions.**

**DevOps Question #1 – Teamwork**

Provide an example of your greatest technical accomplishment where you worked together with a team to solve a difficult problem?

- What were the challenges?
- What was your role?
- How did you put the team ahead of yourself?
- How did you gather information?

**Explanation:**

At the heart of DevOps is an unselfish spirit to **deliver better services**.

There are no **individuals** working on their own agenda.

This idea is easier said than done because working independently and unsupervised is what we are taught is the best thing for good employees.

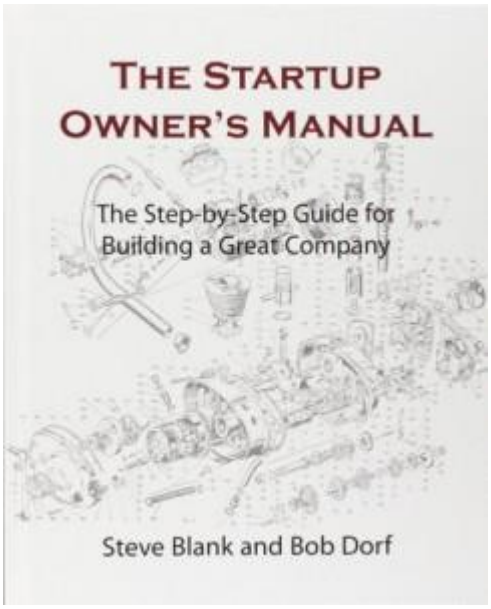
However, DevOps brings the problem solvers together to create a **mastermind** where more people working together are wiser than individuals working alone. Sounds good, right? It's not that easy.

**DevOps interview questions should be crafted to find creative candidates who can work well with others.**

**DevOps Question #2 – Flexibility**

Tell me about a time when you were given instructions to do a task and before you finished, you were asked to change direction more than 5 or more times?

- How did this make you feel?
- What were some of the comments you had (or thought)?
- Did the changes make sense in the end?
- Did you finish the task?




**Explanation:**

If you haven't read the book (or at least chapter one) of The Startup Owner's Manual by Steve Blank and Bob Dorf, you should so you understand the term **Pivot**.

Pivots are changes of direction that happen as often as necessary while the solution to a problem is being determined.

The purpose of pivoting is to avoid spending valuable resources (time, people, money) on something that isn't working.

In other words its failing-fast, then pivoting and moving on.

People who struggle with pivoting will struggle with DevOps, too. Once again being unselfishness should triumph over ego.

Your interview questions (or screening) should help identify people who can pivot or change direction quickly without a lot of fuss.

**DevOps Question #3 – Fill In the Blanks**

Explain a time when you were given a task to figure out a solution for a problem and you had little or no information?

- What type of conversations took place to **fill in the blanks? And with whom?**
- Were there any problems between team members on what was the better solution?
- Were you able to solve the problem in a reasonable time frame with a solid solution that worked?
- How did you contribute to the overall result?

**Explanation:**

Many traditional IT people are notorious for needing to know far too much information before they can get started. Our brains are just wired like that. (We're like Johnny-5. “Input, I need input!")

In fact, gathering requirements before a project starts can take months if you let it.

The goal of DevOps is to create a **mastermind** that can fill in the DNA gaps when there are bits of information missing. This is why I harped so much in the previous chapter about communication and people skills.

Finding people who bend easily and don't get hung-up when they are asked to do task without understanding the full picture is isn't easy.

DevOps is about continuous improvements which come in iteration or small bits at a time. Sometimes requirements come in bits, too.

Your interview questions should help identify people who can work well without knowing the entire stack of requirements up front. Find people who can figure it out. People who are creative thinkers.

## Retrospective (The End)

If you don't understand what I'm saying about the mindset, read this chapter again because it's important.

Let's review who you're looking for:

- First, the candidate needs to be a team player (**no** grand-standers).
- Next, they need to be flexible or willing to change direction without notice (pivot).
- And finally, they need to be able to operate with little or no information, (fill in the blanks).

I don't recall where I read or heard this [maybe on DevOps Cafe podcast] but the best definition of DevOps was actually by Bruce Lee:

***"Be like water!"***

Find people who are unselfish and can adapt quickly to anything they are assigned...better yet, who aren't waiting to be assigned.

## Ready, Set, Go!

I want to thank you for your time. I enjoyed writing this eBook and I hope it will help you to kick-start your plan for instituting DevOps in your organization.

One more thing I want to do before I close is give KUDOS to all my IT brothers and sisters who I know work hard and long hours doing a difficult job.

In no way or form was anything I said to degrade or disrespect you. You are awesome!

As we covered in the beginning, this eBook gets you from A - C, and as you can see that covers quite a bit. Good luck and please leave your comments and feedback. Also, feel free to reach out to me on LinkedIn.

DevOps is learning to be NOBLE and understanding the result is more important than the task!

## Bonus content

Here's some interesting statistics I heard while listening to Jeff Brown's Read to Lead Podcast when he interviewed Kary Oberbrunner, author of **Day Job to Dream Job: Practical Steps for Turning Your Passion into a Full-Time Gig**.

- 86% of US workers feel stuck and are looking for another job.
- 70% of American workers are stressed and experience illness from their current job.
- 30% Feel they will burn out in the next 3 years.
- Heart attacks and injuries are greatest on Monday morning.
- More people die Monday morning around 9AM, than any other day.
- Suicides are highest on Sunday nights from dreading going to work Monday.

***Guard your heart above all else, for it determines the course of your life. - Ancient Proverb***

# About DevOps eBook

**How-To Guide for Building a DevOps Start-Up Plan**

**Joe Sanchez**

The contents of this eBook are for informational purposes only.

In “How-To Guide for Building a DevOps Start-Up Plan,” I share many of the lessons I've learned as an Operations Manager over the last 5 years.

By no means do I assume to know it all, nor am I suggesting my recommendations will work for everyone, or in every IT environment. One thing I am certain of is all IT cultures are different, and therefore require different approaches.

It’s also expected anyone reading this guide will use common sense and do their own due diligence researching this subject before taking any action.

The views, opinions, comments, and suggestions in “How-To Guide for Building a DevOps Start-Up Plan" are my own.



**How-To Guide for Building a DevOps Start-Up Plan**

Version 2.4.4

Read this eBook online at DevOpseBook.com

To find more eBooks by Joe, visit www.VMinstall.com

Disclosure: Some links on this document are Amazon affiliate links.