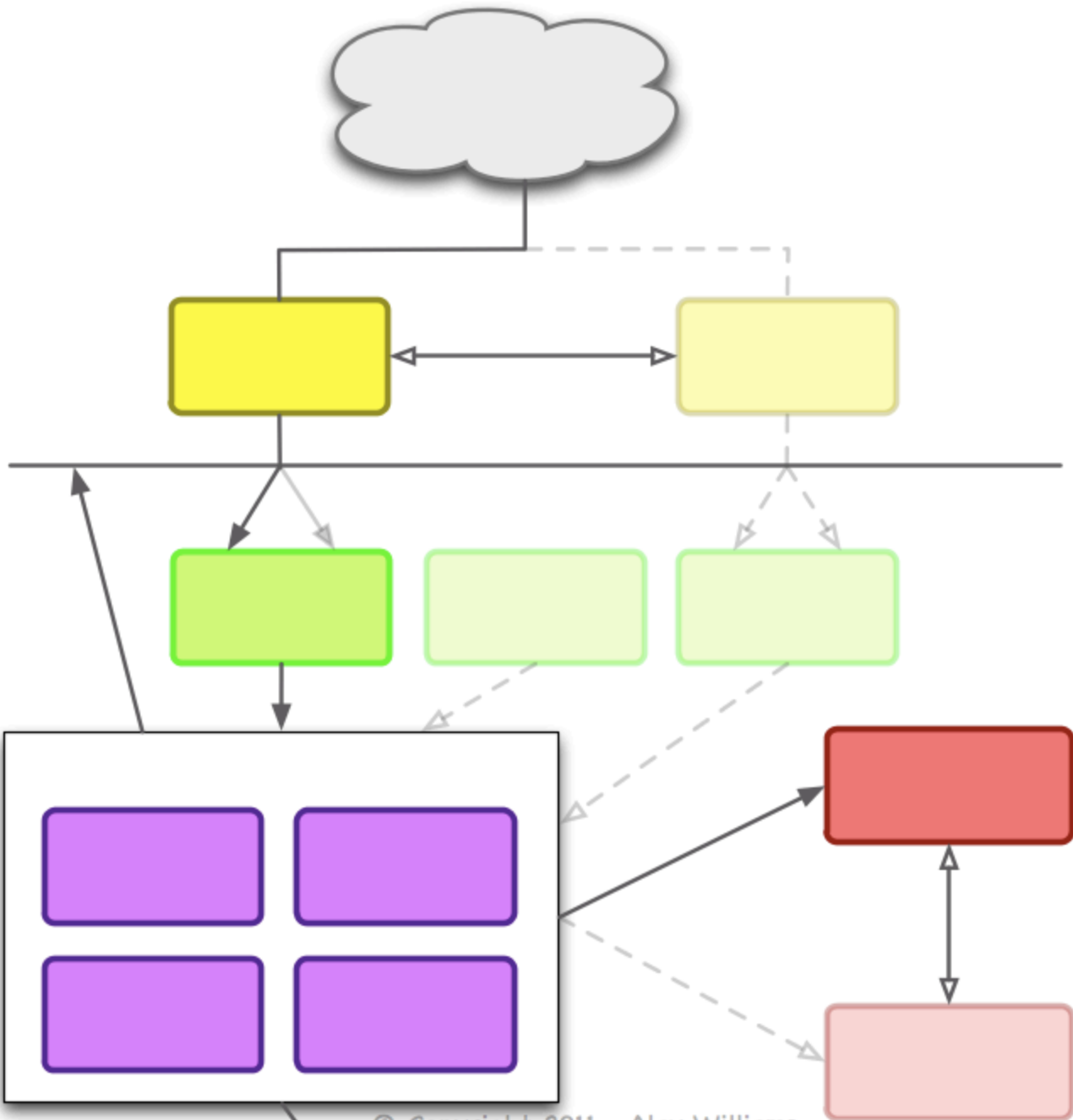
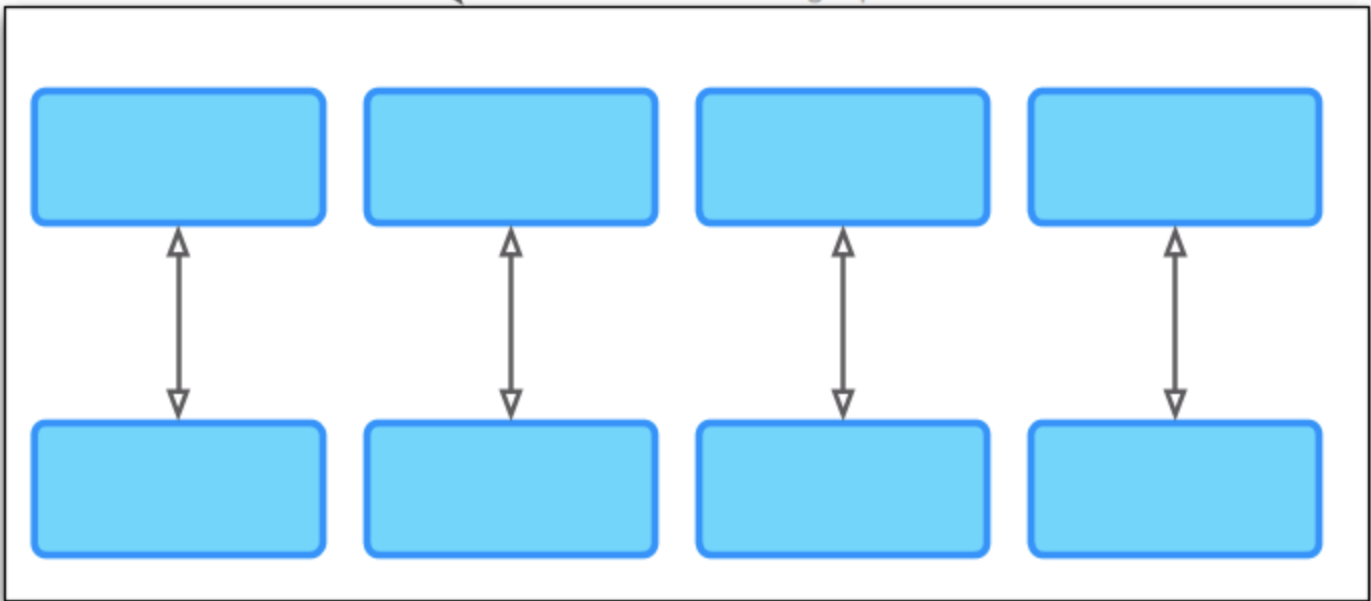


How to get a job at GitHub



© Copyright 2011 - Alex Williams
alexwilliams.ca - scalingexperts.com



by Alex Williams

How to get a job at GitHub

Copyright

How to get a job at GitHub by Alex Williams is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License

Please go here to view this license: <http://creativecommons.org/licenses/by-nc-nd/3.0/>

Published in Quebec, Canada by Alex Williams, www.AlexWilliams.ca

July 2011, First Edition, v1.0

ISBN: 978-2-924017-05-0

Website:	http://www.alexwilliams.ca
Twitter:	@alexandermensa

DISCLAIMER

All trademarks are used with respect to their owners and are designated in *italics* wherever possible. We claim no affiliation with any of the people, software, companies or trademarks listed in this eBook, except for our own.

While every precaution has been taken in the preparation of this eBook, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein. Please forgive us if some examples don't work as intended. We will try our best to update this eBook with corrections and provide purchasers with a FREE updated version of this edition upon request.

Please visit our website for information on updates and revisions to this eBook.

Table of Contents

- Copyright
- Table of Contents
- Acknowledgements
- What the heck?
- Introduction
- Job requirements
- eBook layout
- 1. GitHub's architecture
 - 1.1. The Gist
 - 1.2. Diagrams
- 2. Software you should know
- 3. Ideas for the future
- 4. Fix your life
- Conclusion

Acknowledgements

I want to thank [GitHub](#) for having such an awesome website and customer service. This book is dedicated to you guys.

What the heck?

I don't work for GitHub (yet). In fact, the only association I have with them is an [account](#) on their system for which I keep resetting the password because I have a terrible memory and rarely ever login. Although I sometimes publish code and patches, it's still quite rare since I've signed a million NDAs with all my previous employers, preventing me from owning any copy, distribution or publishing rights to my work.

So who am I and why did I write this book? My name is Alex Williams and I have a website which I rarely update: <http://alexwilliams.ca>. I honestly had nothing better to do when I wrote this. It was either this, or spend the day laying on the beach. Sure `beach > writing` but hey I've been funemployed for half a year now, so spending one day sitting on the back porch under an umbrella on a lovely sunny day typing on my MacBook Air won't hurt me.

Introduction

Here's a story for you:

It was 9am and I had just woken up. Clockwork. Carpe diem suckaz!

I don't have a job, so what's with the fixed morning schedule? Why not? If I were a lazy ass sleeping all day and playing Xbox all night, I wouldn't be awesome and I wouldn't have had the brilliant insight and timing to write this book.



Tip

If you're looking for a job, you need to get out of bed and stop being lazy.

My friend Thierry informed me about GitHub's job opening. I always check my email and Twitter in the morning since I sleep with my BlackBerry under my pillow. Generally my eyes are half-open and sticky, and I usually have dried-up drool on my cheek. Regardless, Thierry's email was short and sweet, therefore completely comprehensible:

| *Trouvé pour toi! (translation: Found for you!)*

Upon reading the job description, I figured since I was familiar with all the tools they use, it would be as easy as buying a 1-way plane ticket to SFO. WRONG! Reading blog posts from *mojombo* (Tom), *defunkt* (Chris), and *Anchor*, you'll quickly realize this job is for a very talented, knowledgeable and experienced Sysadmin.



Warning

If it looks too good to be true, jump on it. At worst you'll learn from your mistake.

The truth is, I'm a little blazé in regards to being a Sysadmin. I've been there done that, hate the default SMS notification sound on my BlackBerry, and took my fair share of yelling and shaky hands at 3am after being told: "*We're losing a million dollars every hour*", so for me it's more of a "*what's next?*".

Personally, I have little to no interest in working at GitHub as a Sysadmin, which is partially what prompted me to write this book. On the other hand, I am extremely interested in an architecture design and future-proofing position, essentially **I want Anchor's job**, minus the waking up at night to fix stuff.

As for you, young cauliflower, I'll help you get that Sysadmin job at GitHub. If you don't get it, then it's your fault not mine. Read the disclaimer at the top of this book.

Job requirements

You see, to be a great Sysadmin at a company like GitHub, you need the following traits:

Lots of skills:

- ✔ Coding skills
 - any programming language, as long as you can learn a new one really quickly
- ✔ Sysadmin skills
 - experience duh! if you're fresh out of school and never saw a load-average over 100, SOL
- ✔ Networking skills
 - managed switches, datacenter stuff, Cisco gear, no certification peddling required
- ✔ Version control skills
 - if you've switched from CVS to SVN to GIT in the last 10 years, you'll have this
- ✔ Waking up late to troubleshoot insanely crazy problems skills
 - don't worry, this can be acquired
- ✔ Adaptability skills
 - things change, you should too
- ✔ Writing skills
 - NO ONE wants a slob who can't clearly document his work

Lots of knowledge:

- ✔ Knowledge of every Linux command on the planet
- ✔ Knowledge of monitoring and profiling tools
- ✔ Knowledge of virtualization stuff
- ✔ Knowledge of load-balancing stuff
- ✔ Knowledge of redundancy stuff
- ✔ Knowledge of the difference between load-balancing and redundancy
- ✔ Knowledge of database stuff
- ✔ Knowledge of file serving stuff
- ✔ Knowledge of automation stuff
- ✔ Knowledge of stuff you don't even know exists

Other stuff:

- ✔ Ability to read faster than you speak
- ✔ Ability to speak at least 4 languages, heck if I can do it, so can you! (ok that's just me showing off, but still)
- ✔ Your own GitHub account, even if you don't push commits due to lengthy NDAs from your previous employers
- ✔ Long hair (haha just kidding, please cut your hair)

If you're looking for a job at GitHub to manage their architecture: servers, switches, routers, and in-house applications, then make sure you see yourself in that list.

Side-note about GitHub

Assuming you know what GitHub is, you should firstly, and most importantly, **know how to use GIT**. You might laugh at me for pointing out the obvious, but if you don't know what your employer does, or why, or how, or whatever, then you're an idiot and you should not read this book. Seriously.

I don't think you need to be a git-guru, considering GIT has a million command-line arguments, that might be impossible even if your first name is Linus. You should at least know the basics in regards to committing, cloning, pulling, pushing, branching, tagging, rebasing, and perhaps a few other fancy things.

eBook layout

This eBook layout is fairly simple, considering I gave myself the goal to write it in just 1 day, on the most beautiful day of the summer.

Chapter 1 has lovely diagrams describing GitHub's entire internal architecture from servers to software.

Chapter 2 will list every piece of software they use, at least according to what they've disclosed. It's the meat and potatoes, you know? The things you NEED to know to get this job.

Chapter 3 will provide ideas that you can implement assuming you do get that job. This ranges from virtualizing database servers, to replacing that brilliant but crazy SSH+MySQL patch with something much simpler and more elegant (hehe).

Finally, **Chapter 4** will explain what else you can do to help yourself get this or future jobs at GitHub. So far they've only announced the need for 1 Sysadmin, but eventually they will need more, so I'll try to end the book with some timeless information for you Sysadmin/ninjas out there.

1. GitHub's architecture

Now that we've got the fluff out of the way, we'll discuss the internal architecture of GitHub.

First, since you're not lazy you'll want to read the following blog posts:

- <https://github.com/blog/530-how-we-made-github-fast>
- <http://www.anchor.com.au/blog/2009/09/github-designing-success>

Realize these were written in October 2009, which was almost 2 years ago. This means a LOT of things have changed and you'll need the ability to adapt to whatever they throw at you on your first day.

1.1. The Gist

Here's what I found so far:



Note

I've intentionally omitted certain details to prevent malicious losers from attacking GitHub's network. I greatly respect what GitHub has done for the community so far and wouldn't want to make it too easy for people to start any wars against them.

- ➕ Github.com is only accessible over IPv4
- ➕ Rackspace has assigned them a /27, which means technically 30 useable IPs
- ➕ Two of those addresses point to identical edge routers/switches, which are on Rackspace's network
- ➕ The website is only accessible over HTTPS, arriving at www.github.com is 301 redirected to github.com
- ➕ The multi-tiered architecture allows for failover of any component without impacting the user experience

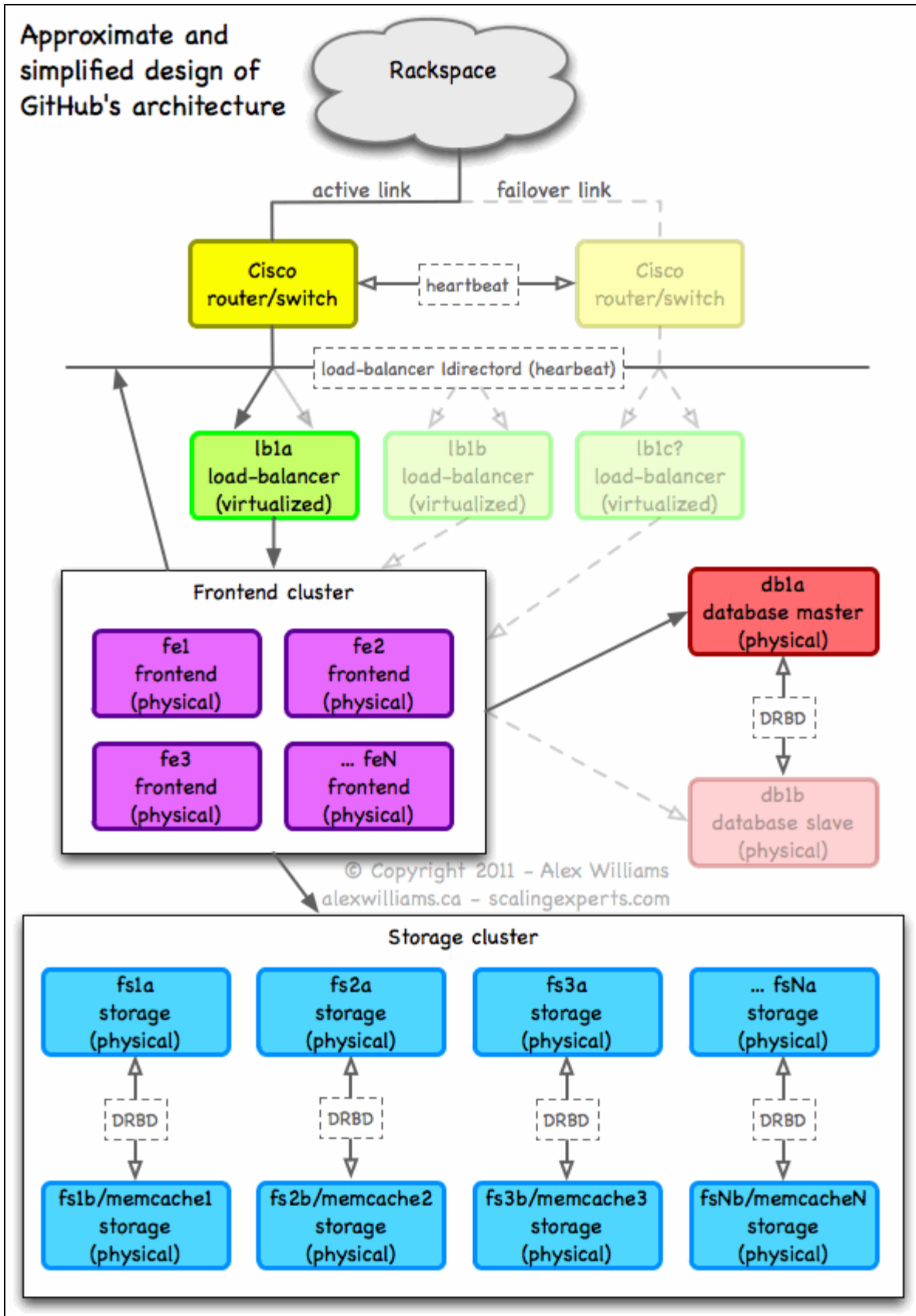
1.2. Diagrams

According to the Sysadmin job posting from July 6th 2011, GitHub is running exactly 43 dedicated servers at Rackspace. Not bad, but nothing to scare an experienced sysadmin (unless there's no documentation).

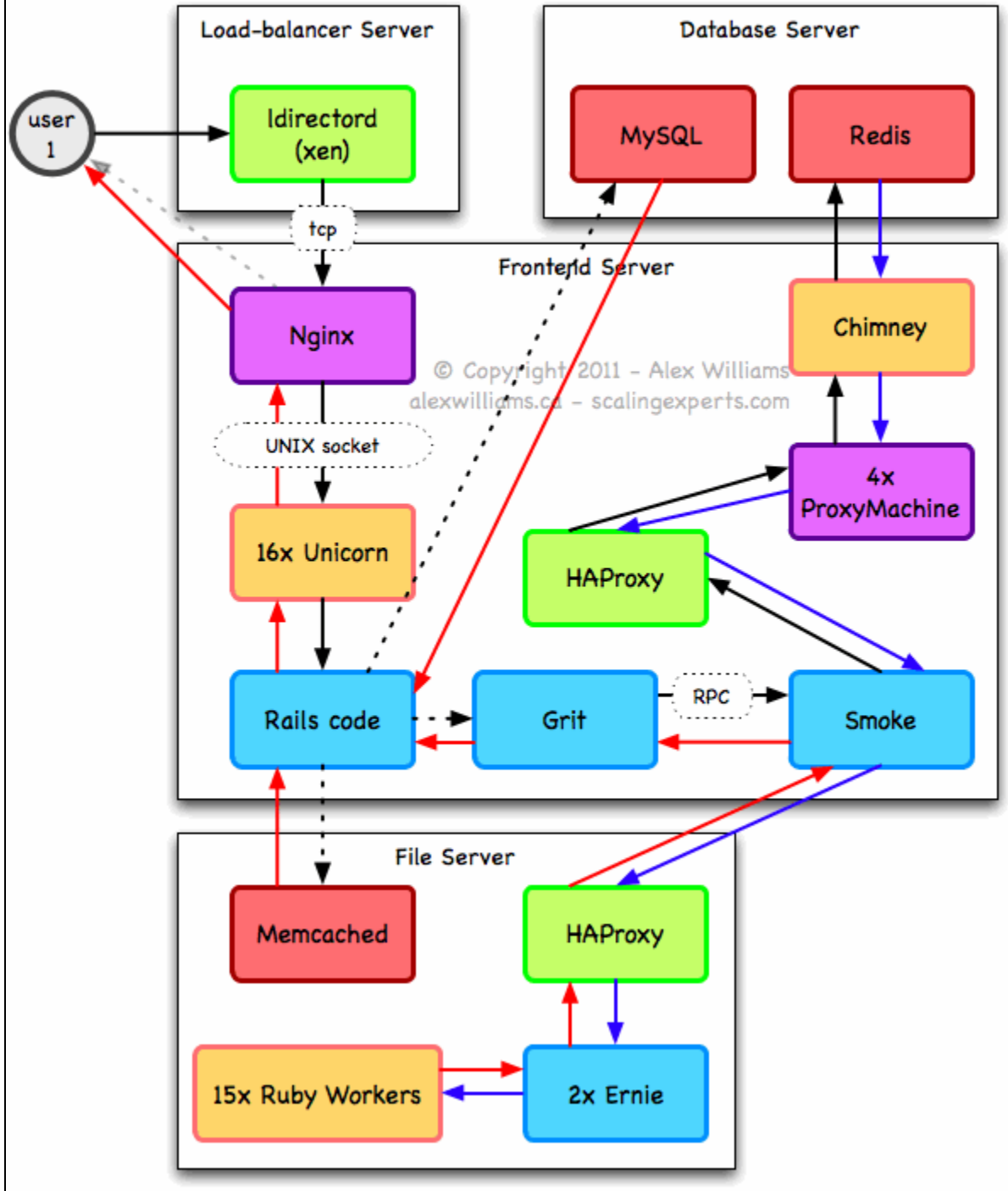
Here's some mock drawings of what I **think** their architecture resembles. I could be completely wrong since I obviously don't work there (yet), but it's the best I can do based on their various blog posts and non-WikiLeaked information.

Regardless, the goal is to have a rough idea of what you're getting yourself into by even applying for the Sysadmin job. Even if you don't feel comfortable with all the software and hardware they use, it doesn't mean you can't learn that stuff. So please take these examples as guides as opposed to absolute references.

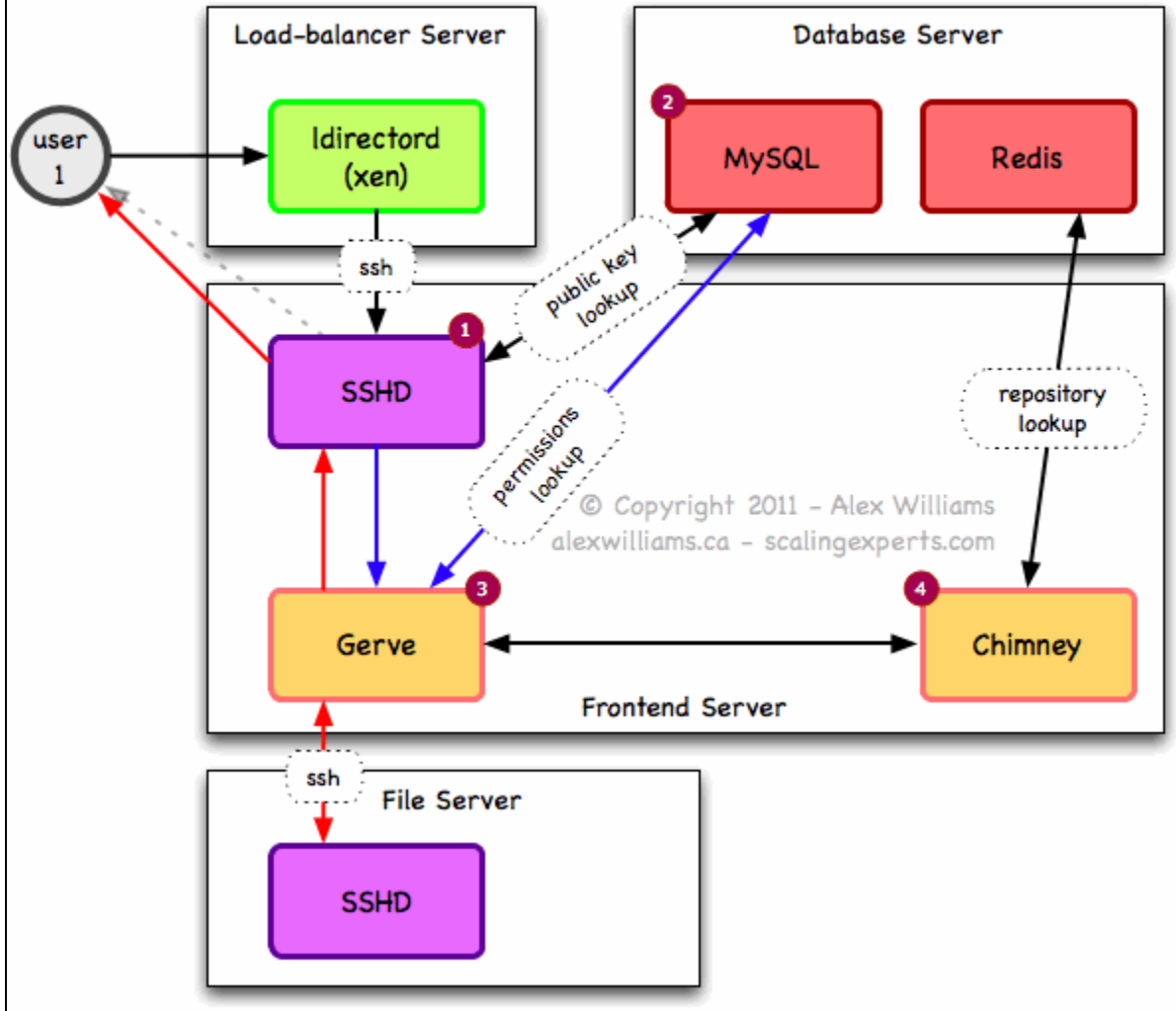
Approximate and simplified design of GitHub's architecture



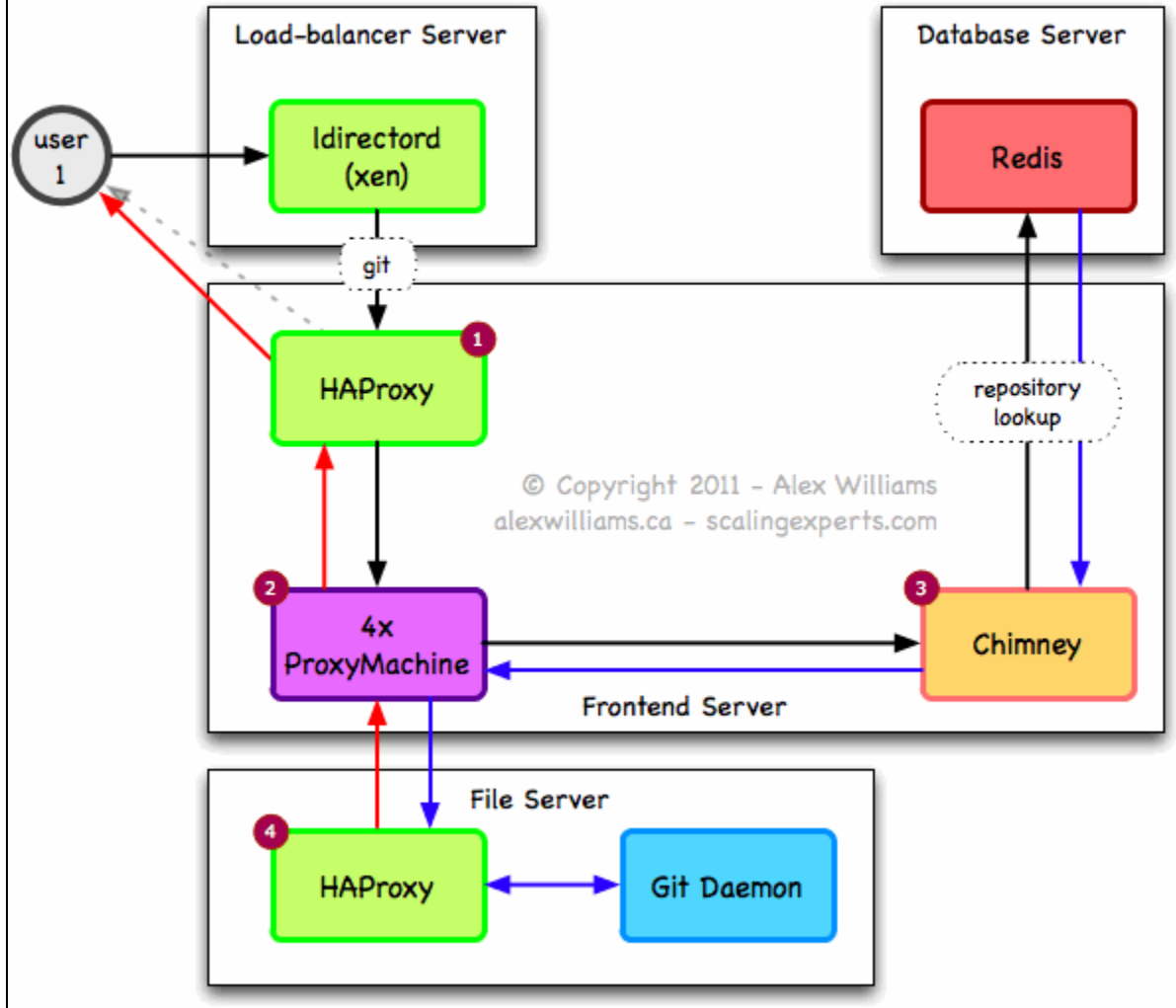
Approximate and simplified design of GitHub's HTTP requests



Approximate and simplified design of GitHub's SSH requests
















Approximate and simplified design of GitHub's GIT requests



2. Software you should know

I'm not going to describe in detail each piece of software, but just be sure you're familiar enough with everything on this list, and other things not on this list:

-  Idirectord (does load-balancing)
-  heartbeat (checks status of load-balancers)
-  Nginx (serves static files, rewrites URLs and maybe SSL offloading)
-  Unicorn (Rails/Rack worker process)
-  Ruby (programming language)
-  HAProxy (also does load-balancer)
-  Memcached (cache without persistence)
-  Redis (cache with persistence)
-  MySQL (database)
-  Xen (virtualization layer)
-  GIT (also common?)
-  Puppet (automating configurations and deployment)
-  Munin (graphs of your server's activity)



Information

Also note, even if you're not familiar with ProxyMachine, Chimney, Grit, Smoke or Ernie, these things should be picked up pretty quickly, so don't worry about it.

You will also likely need to be familiar with everything listed in the job description, so don't be lazy.

In any case, if you want to be a Sysadmin at GitHub, other than having to know everything, your ability to learn quickly, understand concepts and apply them intelligently will help you much more throughout your career as opposed to mastering only 1 specific software and calling it a day.

3. Ideas for the future

So you landed a job interview at GitHub? Congrats! But guess what, I'm convinced your job description will change within the first 6 months of working there. In any case if it doesn't then you'll probably be bored to death.

If you want to be hired, you'll need to use your brain a little, but preferably a lot. You need to provide VALUE to your future employer. If you're just a cog who works and then goes home, you're useless. You'll increase your chances of landing the job if you can provide them with some great ideas for improving their infrastructure, lowering latency, saving money, or even making more money.

Below is my personal list of ideas for GitHub. You can try using them but remember you're probably not the only person to have read this book, which means they'll know it's not your original idea. Regardless, if you know what you're talking about, it might just work 😊

- 💡 Add IPv6 support on the load-balancers.
 - Assuming the load-balancing software is IPv6 ready (and if it isn't, why don't you write a patch for that?), then you can request an IPv6 subnet from Rackspace and simply let your load-balancers do V6 to V4 translation. Don't think about adding IPv6 across all tiers as this will be practically impossible (and crazy) in a production environment. One step at a time.
- 💡 Add MySQL read-only slaves for generating reports/stats/etc
 - According to one blog post, there's only 1 MySQL database which can be used at all times, because as we know, with DRBD you can only have 1 active server. That seems like a lot of stress on just 1 machine, and a big single point of failure. How about adding read-only slaves for non-essential tasks?
- 💡 Virtualize everything!
 - Yeah I know that sounds crazy, but personally I know it's possible. I know it's not the brightest idea for someone who just migrated **AWAY** from virtualization, but there's a big difference between having 100% of the resources (virtualizing a dedicated server) as opposed to using a VPS service such as Linode. It'll streamline the setup across ALL servers and provide some added functionality such as live migrations and instant deployments. I would also strongly recommend looking into KVM as opposed to Xen and utilize the wonderful features and fantastic implementation of Hardware Virtualization (HVM) provided by KVM. As for disk performance, you can easily do PCI passthrough for native performance, which is simplified when you're using a high-end RAID adapter.
- 💡 Change fileserver storage platform
 - I love DRBD, but it's limited to only 2 DRBD nodes. Also, the fact that it's block-level means you're transferring data that doesn't exist, even zeros (yes I know there's workarounds for that, but still). An evaluation of other data replication/cloning and storage solutions would be nice to ensure even better uptime. It also avoids having servers idling away doing nothing except serving memcached requests. The worst is when DRBD freaks out and corrupts data on BOTH servers. I've seen it happen. Not pretty.

As you can see, there are many things GitHub can do to improve their architecture, and as they grow there will continually be things to fix, change, improve, speed-up, upgrade, redesign and re-architect.

As a Sysadmin, you'll be responsible for making sure things work as they're supposed to, as well as solving problems once and for all (no stupid permanent hacks).

4. Fix your life

So you've made it this far, taken all my advice, learned a bunch of new software and interviewed for the Sysadmin job, but you still didn't get it?

Well, I'm sorry to hear that. We've all failed job interviews, but it's not a reason to put your head down or go back to sleep. You should see it as a learning experience. Take advantage of the fact that you're still alive, and go learn something else. Perhaps a new spoken language, or learn how to take care of kids (trust me that's a challenge & a half). 🙌

I'm going to provide you with a list of things I do almost every day. These things keep me active and they keep my brain from losing too many cells. It has become part of my lifestyle and I think you should also do it.

- ✔ Read a book, tech blogs, news sites, something for your brain. The more you stay connected, the easier it will be to get back into the job market, or even to jump to a new job, or improve your current career choice.
- ✔ Watch a funny movie or TV show. Allow yourself to LAUGH, every day! Your brain needs it.
- ✔ Do something physical. Don't be a bloody nerd drone always in front of your computer. Go out for a run, bike, rollerblade, walk, have sex, whatever! Just be active! If you sit there and let yourself become fat and lazy, well you'll become fat and lazy. That one's obvious.
- ✔ Talk to people. Personally I hate talking to people, especially stupid ones. But you'll be surprised how much you can learn from people no matter how smart or dumb they may be. Everyone has different perspectives and life experiences, so feel free to talk and listen to others. Give them your attention, get their respect.
- ✔ Read MY book. Yes this is a shameless plug, but make sure you purchase and read my eBook: **Web Scaling vol. 1 - Small Architectures**. <http://www.ScalingExperts.com/books>
- ✔ Experiment with the unknown. Along the lines of talking to random people (which can be really weird sometimes), you should not be scared to try new software, take a different route home, and even eat new foods.
- ✔ Enjoy LIFE! If you're not enjoying life, it will show. People don't want to be with people who aren't trying to be as awesome as they can. On the other hand, people love happy people who are doing things, ambitious and just trying to expand their mind and body. Do those things, enjoy life!

Conclusion

I hope you enjoyed this FREE book. Please feel free to distribute, copy, print (be nice to the environment though), share, repost, convert to audio, translate this book. Please don't sell this FREE book, also don't modify it without my permission.

Thanks!

Alex Williams

<http://alexwilliams.ca> - <http://scalingexperts.com>