Servers & Developers

Julian Nadeau
Production Engineer

shopify

Provisioning & Orchestration of Servers

Setting a server up

Packer - one server at a time

Chef - all servers at once

Containerization

What are Containers?

A look at Kubernetes

Developer Environments

Similar to production?

Local vs Remote

Can containers be used?

Servers & Stuff

shopify

Setting a server up

Imagine your job is to setup servers. You want a web-app to run.

- Install a few dependencies, maybe Ruby, PHP, Node.
- Get a web server up and running (maybe Nginx or Apache?).
- Open up a few firewall ports.... and Done!

This is called an artisanally set up server. In server-land, artisanal is generally not a good idea.

Setting a server up (cont)

Imagine you need to set that server up again, will you remember how to? Maybe you'll script it (more on this in a moment).

Now, imagine that you have 10 servers and for some reason you need to update Ruby on all everything, but this causes some issues with another dependency.

Now imagine that with 50 servers, 100 servers, 1000 servers.

Server Stats for Shopify

2

DATACENTRES

500+

WEB SERVERS

70+

JOB SERVERS

+++

DATABASE SERVERS, CACHING SERVERS, LOAD BALANCERS

Setting a server up (cont)

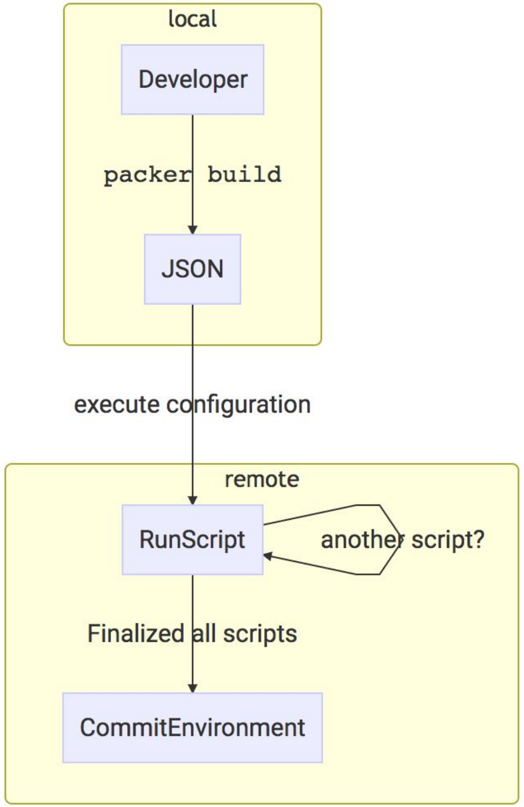
Cool, so now that we’ve established that artisanally crafted servers aren’t the best idea, what can we do?

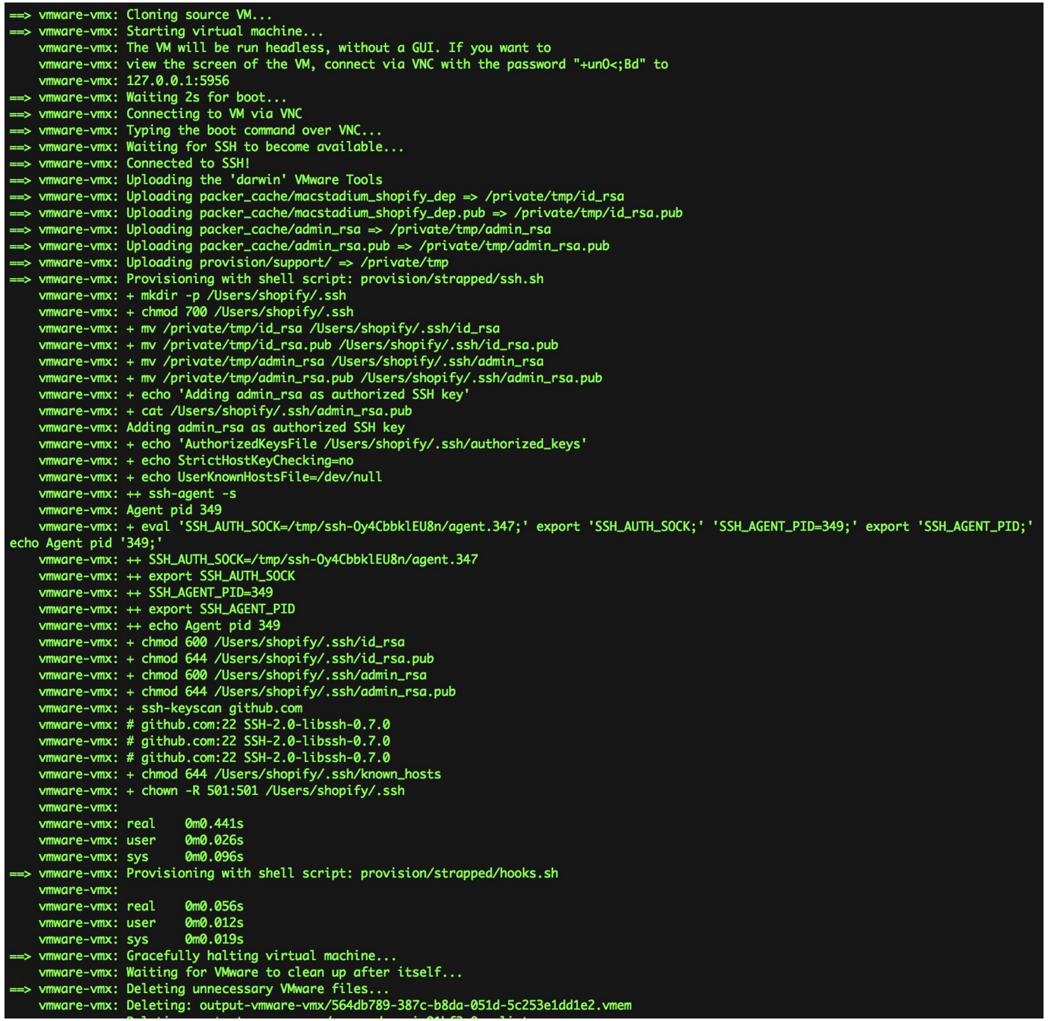
Let’s script it.

Choosing a set of scripts and settings for a server is called a “configuration”. Applying that configuration is called “provisioning”.

We’ll talk about both in the next few slides.

PACKER





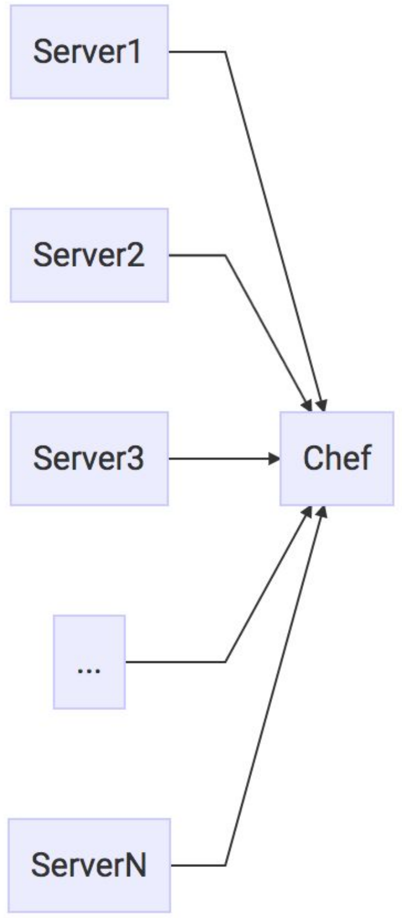
```
==> vmware-vmx: Cloning source VM...
==> vmware-vmx: Starting virtual machine...
    vmware-vmx: The VM will be run headless, without a GUI. If you want to
    vmware-vmx: view the screen of the VM, connect via VNC with the password "+unO<;Bd" to
    vmware-vmx: 127.0.0.1:5956
==> vmware-vmx: Waiting 2s for boot...
==> vmware-vmx: Connecting to VM via VNC
==> vmware-vmx: Typing the boot command over VNC...
==> vmware-vmx: Waiting for SSH to become available...
==> vmware-vmx: Connected to SSH!
==> vmware-vmx: Uploading the 'darwin' VMware Tools
==> vmware-vmx: Uploading packer_cache/macstadium_shopify_dep => /private/tmp/id_rsa
==> vmware-vmx: Uploading packer_cache/macstadium_shopify_dep.pub => /private/tmp/id_rsa.pub
==> vmware-vmx: Uploading packer_cache/admin_rsa => /private/tmp/admin_rsa
==> vmware-vmx: Uploading packer_cache/admin_rsa.pub => /private/tmp/admin_rsa.pub
==> vmware-vmx: Uploading provision/support/ => /private/tmp
==> vmware-vmx: Provisioning with shell script: provision/strapped/ssh.sh
    vmware-vmx: + mkdir -p /Users/shopify/.ssh
    vmware-vmx: + chmod 700 /Users/shopify/.ssh
    vmware-vmx: + mv /private/tmp/id_rsa /Users/shopify/.ssh/id_rsa
    vmware-vmx: + mv /private/tmp/id_rsa.pub /Users/shopify/.ssh/id_rsa.pub
    vmware-vmx: + mv /private/tmp/admin_rsa /Users/shopify/.ssh/admin_rsa
    vmware-vmx: + mv /private/tmp/admin_rsa.pub /Users/shopify/.ssh/admin_rsa.pub
    vmware-vmx: + echo 'Adding admin_rsa as authorized SSH key'
    vmware-vmx: + cat /Users/shopify/.ssh/admin_rsa.pub
    vmware-vmx: Adding admin_rsa as authorized SSH key
    vmware-vmx: + echo 'AuthorizedKeysFile /Users/shopify/.ssh/authorized_keys'
    vmware-vmx: + echo StrictHostKeyChecking=no
    vmware-vmx: + echo UserKnownHostsFile=/dev/null
    vmware-vmx: ++ ssh-agent -s
    vmware-vmx: Agent pid 349
    vmware-vmx: + eval 'SSH_AUTH_SOCK=/tmp/ssh-Oy4CbbklEU8n/agent.347;' export 'SSH_AUTH_SOCK;' 'SSH_AGENT_PID=349;' export 'SSH_AGENT_PID;'
echo Agent pid '349;'
    vmware-vmx: ++ SSH_AUTH_SOCK=/tmp/ssh-Oy4CbbklEU8n/agent.347
    vmware-vmx: ++ export SSH_AUTH_SOCK
    vmware-vmx: ++ SSH_AGENT_PID=349
    vmware-vmx: ++ export SSH_AGENT_PID
    vmware-vmx: ++ echo Agent pid 349
    vmware-vmx: + chmod 600 /Users/shopify/.ssh/id_rsa
    vmware-vmx: + chmod 644 /Users/shopify/.ssh/id_rsa.pub
    vmware-vmx: + chmod 600 /Users/shopify/.ssh/admin_rsa
    vmware-vmx: + chmod 644 /Users/shopify/.ssh/admin_rsa.pub
    vmware-vmx: + ssh-keyscan github.com
    vmware-vmx: # github.com:22 SSH-2.0-libssh-0.7.0
    vmware-vmx: # github.com:22 SSH-2.0-libssh-0.7.0
    vmware-vmx: # github.com:22 SSH-2.0-libssh-0.7.0
    vmware-vmx: + chmod 644 /Users/shopify/.ssh/known_hosts
    vmware-vmx: + chown -R 501:501 /Users/shopify/.ssh
    vmware-vmx:
    vmware-vmx: real    0m0.441s
    vmware-vmx: user    0m0.026s
    vmware-vmx: sys     0m0.096s
==> vmware-vmx: Provisioning with shell script: provision/strapped/hooks.sh
    vmware-vmx:
    vmware-vmx: real    0m0.056s
    vmware-vmx: user    0m0.012s
    vmware-vmx: sys     0m0.019s
==> vmware-vmx: Gracefully halting virtual machine...
    vmware-vmx: Waiting for VMware to clean up after itself...
==> vmware-vmx: Deleting unnecessary VMware files...
    vmware-vmx: Deleting: output-vmware-vmx/564db789-387c-b8da-051d-5c253e1dd1e2.vmem
```

CHEF™

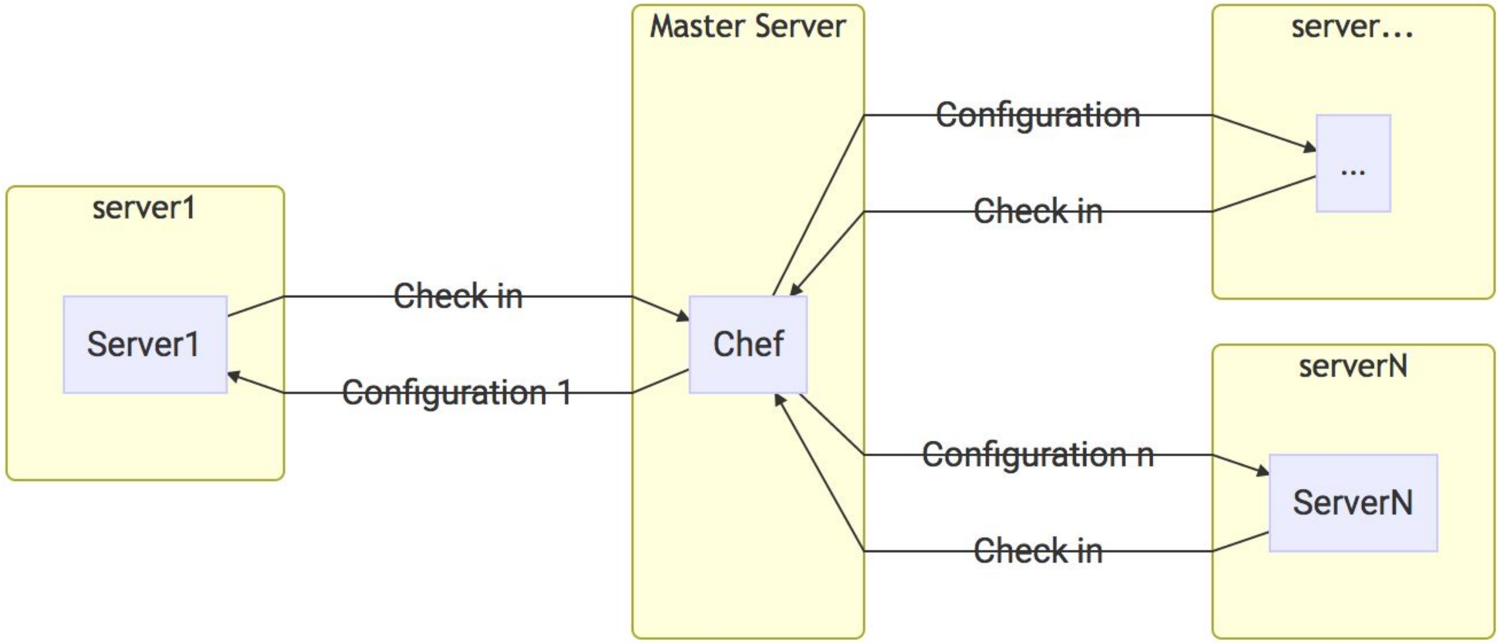
- Packer is a great tool for managing the initial provisioning of a server, but what if you have a fleet of many servers?

- Chef allows you to manage many servers through a centralized “Chef Server”. The master server tells the provisioning node what scripts and programs it should run, and allows for iteration to happen

- The provisioning node checks in on a schedule to see if anything new is needed



CHEF™

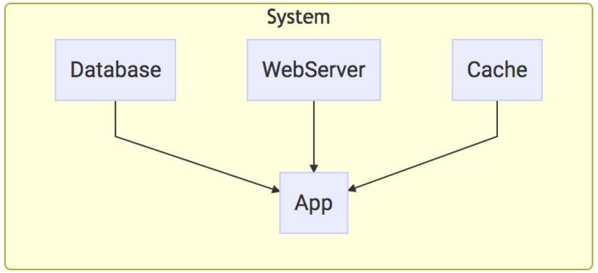


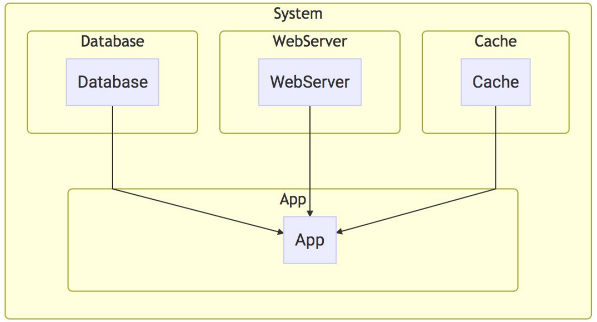
Master Server
Chef
server1
Server1
Check in
Configuration 1
server...
...
Configuration
Check in
serverN
ServerN
Configuration n
Check in

Containerization

What are containers?

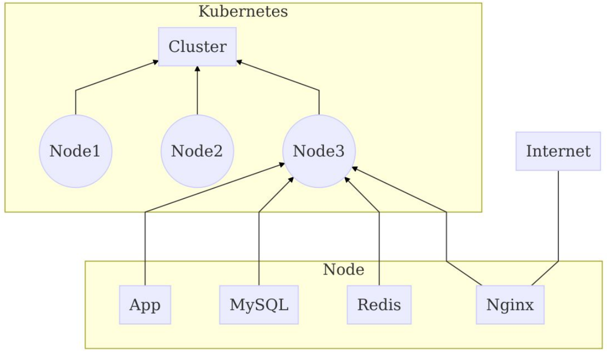
- Containers are a method of operating system virtualization that allow you to run an application and its dependencies in separate “groups”.
- This means that we can basically group off a system into “containers” and have many things running at the same time.
- Without containers, processes can fight for resources (memory, CPU, etc) or run one service per computer.
- With containers, processes can be pre-allocated a set amount of resources so we can run many per host without the worry of them competing. This is not enabled by default however.



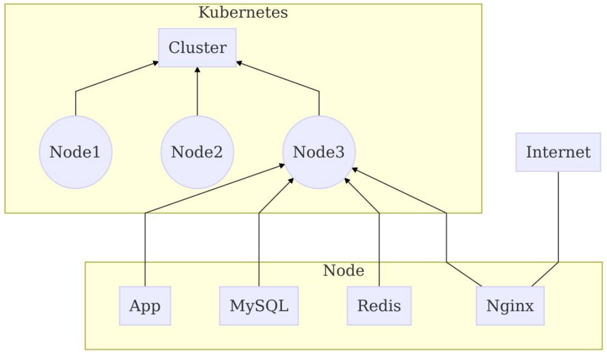


What are they good for?

- As mentioned, we can restrict resource usage so we don't have one thing taking everything up
- Since the containers are pre-built images, we get consistency despite just running the image
- We have different versions of container images, so if something breaks we can revert to using an older image
- Since things are grouped, we don't have to worry about dependencies conflicting (perhaps your database requires one version of C, while your cache requires a different version).



- Kubernetes is a tool from Google that allows us to manage containers without caring too much about the servers.
- We have a “cluster” (collection) of “nodes” (servers) where we run our application.
- “Pods” (a set of containers) run selected list of “services” (database, cache, webserver etc).
- For example, we may see a simple app run a database container and webserver in the same pod.
- For those that want a deeper look on their own, this is an abstraction of *Borg* and *Omega* in Google’s internal container orchestration.



- Kubernetes is told to run a system as you define it. This means that if a container dies, it will be brought back up somewhere else. It also means that it will kill something that is no longer in your system definition.
- This means that you no longer need to worry and freak out if a specific server dies as Kubernetes will just give the workload to another server instead.
- This allows us to "treat servers like cattle" (aka they're all replaceable and are a means to an end)

Developer Environments

shopify

Similar to Production

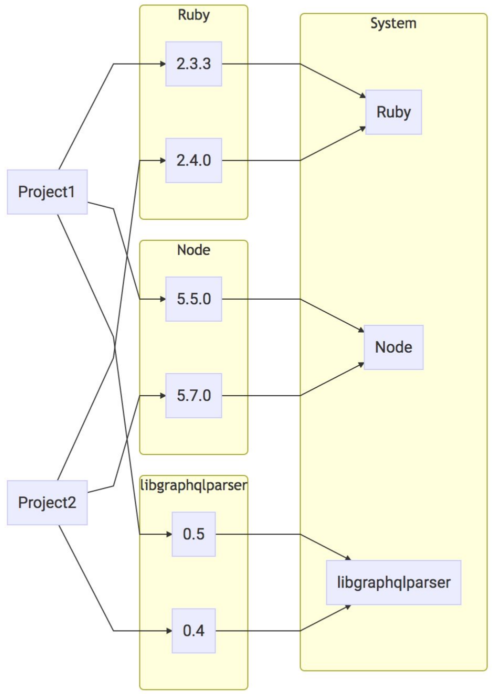
- Developer environments are similar to production server provisioning as they have to install and maintain dependencies as well

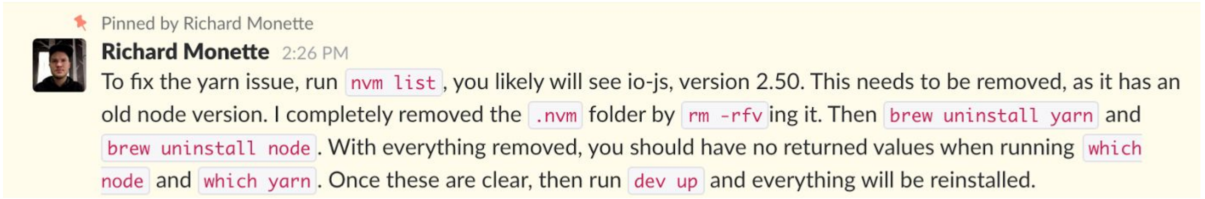
- The main difference is that we may need multiple versions of the same dependency depending on the project (i.e. Ruby 2.3.3 vs Ruby 2.4.0, iOS 9 vs iOS 10, etc).

- For many cases, a system can only have one version that is active at a time.

- We also don't know the state of the system as people may have installed custom environment-impacting programs, scripts, profiles, etc

- This is an example for 2 projects, 3 dependencies, 2 versions per dependency. In reality, there will be many more projects, many more dependencies, and many more versions.

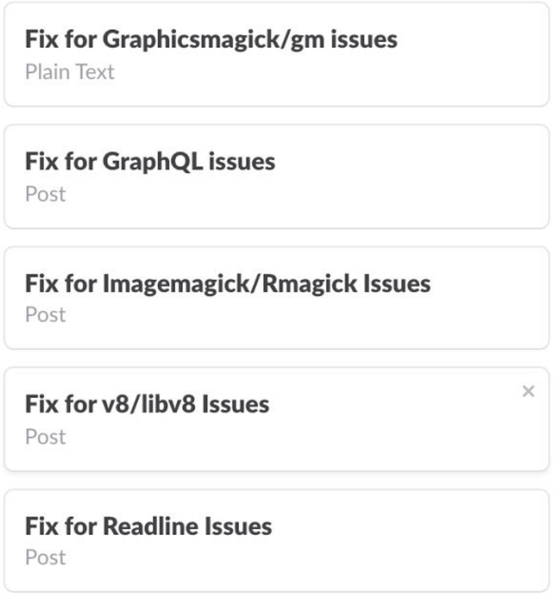




Pinned by Richard Monette

Richard Monette 2:26 PM

To fix the yarn issue, run `nvm list`, you likely will see io-js, version 2.50. This needs to be removed, as it has an old node version. I completely removed the `.nvm` folder by `rm -rfv`ing it. Then `brew uninstall yarn` and `brew uninstall node`. With everything removed, you should have no returned values when running `which node` and `which yarn`. Once these are clear, then run `dev up` and everything will be reinstalled.



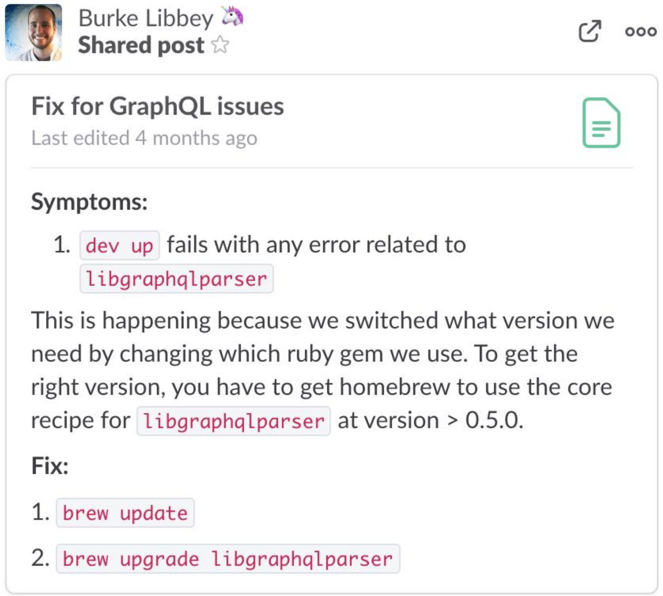
Fix for Graphicsmagick/gm issues
Plain Text

Fix for GraphQL issues
Post

Fix for Imagemagick/Rmagick Issues
Post

Fix for v8/libv8 Issues
Post

Fix for Readline Issues
Post



Burke Libbey 🦄
Shared post ☆

Fix for GraphQL issues
Last edited 4 months ago

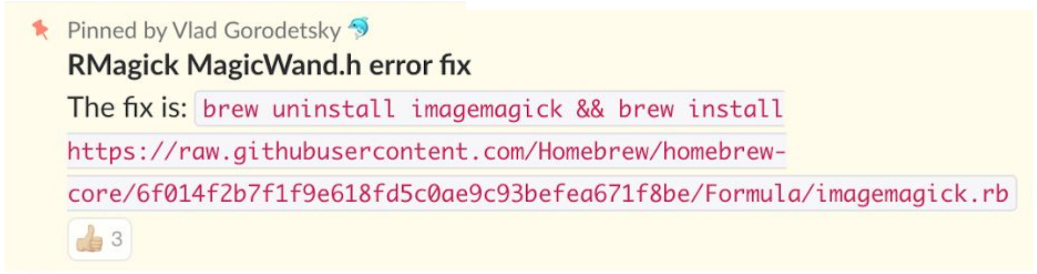
Symptoms:

1. `dev up` fails with any error related to `libgraphqlparser`

This is happening because we switched what version we need by changing which ruby gem we use. To get the right version, you have to get homebrew to use the core recipe for `libgraphqlparser` at version > 0.5.0.

Fix:

1. `brew update`
2. `brew upgrade libgraphqlparser`



Pinned by Vlad Gorodetsky 🐬

RMagick MagicWand.h error fix

The fix is: `brew uninstall imagemagick && brew install https://raw.githubusercontent.com/Homebrew/homebrew-core/6f014f2b7f1f9e618fd5c0ae9c93befea671f8be/Formula/imagemagick.rb`

👍 3

Local vs Remote

- Can work offline, therefore can code anywhere (like an airplane)
- Dependencies can clash between projects.
- Low barrier of entry for text editors and IDEs.
- Low latency, the file system is local to it is quick to update.

- Separate environments are possible (one server per project).
- Mitigates the dependency hell.
- You have to have a remote-compatible IDE/text editor and always be online, or be SSHed into a server at all times.
- File system is not local, so syncing errors can occur

Containerizing Dev Environments?

- We’ve been talking a lot about environments conflicting with each other, whether by dependency or resource (cpu, ram, etc)
- If we could containerize applications and their dependencies, we would be able to solve a lot of these problems.
- Currently, containerizing makes the code also be separated to a different system (in the container), which means we have to SSH in (giving us the problems of remote developer environments)
- This is a problem we’re currently thinking about, we have not solved it. MiniKube is a potential candidate (let’s you run Kubernetes locally) that is showing promise.

Thanks!

@jules2689
julian@shopify.com

shopify