GETTING STARTED WITH CF'S DOCKER IMAGES

Charlie Arehart, Independent Consultant
CF Server Troubleshooter
charlie@carehart.org
@carehart (Tw, Fb, Li, Slack, Skype, GitHub)

Updated Sep 30, 2019

Wifi Access

AdobeSummit2019

Adobe2019
(case-sensitive)

- As session description made clear:
  - Getting started with Docker
  - Getting started with Adobe's CF Docker images

WHAT WE WILL BE TALKING ABOUT TODAY

Charlie Arehart
CArehart.org
@carehart

- May be a developer, admin, tester, team lead, business person, or other
- As for your background with Docker, I am suspecting either:
  - You may never had heard of docker (and just came out of curiosity)
  - Or have heard of it but never really understood it
  - Or tried to install it, had trouble and gave up
  - Or got it working but never used it with CF images
  - Or only have used the Ortus Commandbox (or Lucee) CF images
- I'm trying to speak to all these audiences today, so bear with me ☺

WHERE YOU MAY BE COMING FROM

Charlie Arehart
CArehart.org
@carehart

WHO AM I?

- 20+ years in CF, 35+ in enterprise IT
- Independent consultant, providing server troubleshooting
- Active in the community (forums, portal, mailing lists, social media)
  - My contact info is on front slide

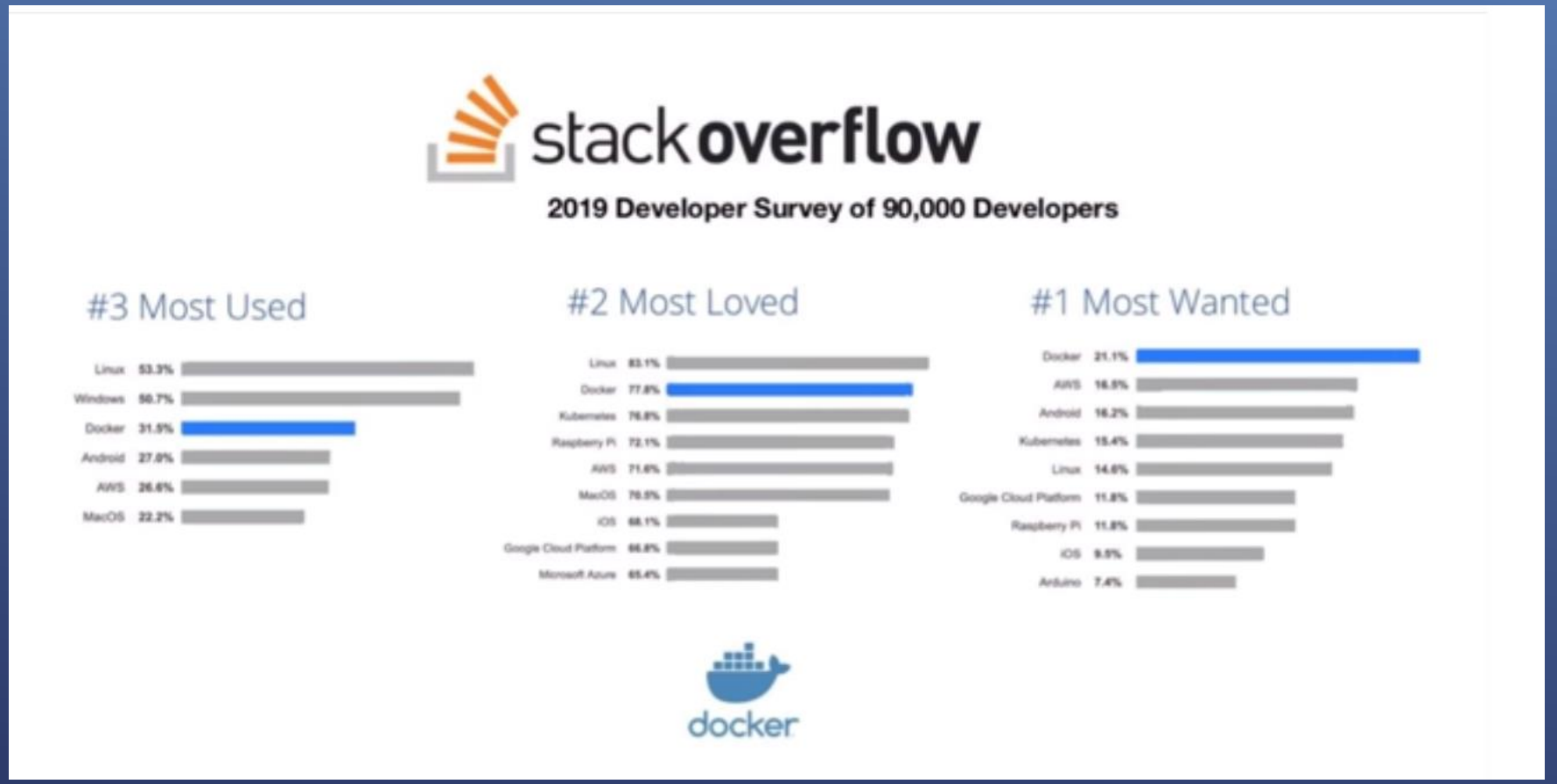
- Slides will be available at carehart.org/presentations

Charlie Arehart
CArehart.org
@carehart

DID YOU KNOW THERE EVEN ARE ADOBE-PROVIDED CF IMAGES?

- …for those already familiar with docker, I mean here
  - You'd be forgiven for not knowing. They don't get much press
- For a few years the Ortus-provided CommandBox image has existed
  - Most resources on the web regarding CF and Docker refer to them
- Adobe's are not listed at dockerhub (the default "registry" for Docker)
  - Instead they are at bintray.com
- Let's take a look at the site

Charlie Arehart
CArehart.org
@carehart



WHO'S EXCITED ABOUT DOCKER?

Charlie Arehart
CArehart.org
@carehart

- Let me run some examples
- Will explain more later what I did, in more detail

CAN I EXCITE YOU ABOUT DOCKER?

Charlie Arehart
CArehart.org
@carehart

HOW EASY IS IT TO INSTALL DOCKER?

- To run Docker, you do need to install it
  - There are installers for Linux, Windows, and MacOS
  - For Windows Home and MacOS before Sierra, must use Docker Toolbox option
    - Those on Windows 10 Pro and above, or above MacOS Sierra, use Docker Desktop
  - Install just takes minutes, and once done can run "docker' commands…

- Let's do that, with available "hello world' image: *docker run hello-world*
  - Will be downloaded if not run before
- Shows "hello" message, confirming that all is well
- What happened?
  - Docker checked to see if I had the image already downloaded
  - If not, it downloaded it from the default "registry", dockerhub.com
  - A "container" for the image was created, then run, and it stopped

YOUR FIRST DOCKER IMAGE

Charlie Arehart
CArehart.org
@carehart

- For testing out Docker/playing with it
  - Free site: play-with-docker.com
- For development
  - Many cloud platforms let you run docker (via "container services") at low-cost
- For production
  - Such cloud providers also offer "orchestration" of containers, such as via Swarm, Kubernetes, as we will discuss again later

HOW TO USE IT WITHOUT INSTALLING IT

Charlie Arehart
CArehart.org
@carehart

WHAT IS DOCKER

- Relatively new platform, released as open source 2013
- Helps you build, ship and run applications, pretty much anywhere
  - Across disparate platforms, OS's, bit -levels, processors and more
- Becoming prevalent in IT as an alternative way to run and deploy server software
- Likened to vm's (not a great analogy)
  - Better to think "packaged application" (not itself really new with Docker)

Charlie Arehart
CArehart.org
@carehart

- Changes the normal approach: rather than install, just pull/run
  - Remove when done or if no longer interesting (will see how, later)
  - Can also preserve data you may create with/within the container (more later)
- Important: most Docker images are usually Linux
  - But Docker for Windows or MacOS lets you run them!
  - That's one of the game changers
- Allows you to explore things you might normally not have considered
  - Let's see some examples

EXPLORING NEW SOFTWARE VIA DOCKER

Charlie Arehart
CArehart.org
@carehart

- Can be used to run server software you might usually work with
  - Web servers (like Apache, IIS, and nginx)
  - Databases (like MySQL, SQL Server, Postgres, and more)
  - Nosql datastores (like couchbase and Mongodb)
- And also such things as:
  - Reverse proxies (like haproxy and Træfɪk)
  - Caches/accelerators (like memcached and Varnish)
  - Search engines (like Solr and ElasticSearch)
  - Messaging brokers (like rabbitmq and rocketmq)
  - and more

DOCKER IMAGES FOR NEARLY ALL SERVER SOFTWARE

Charlie Arehart
CArehart.org
@carehart

- Docker images also exist for application server software (as alternatives to ColdFusion) such as:
  - Tomcat
  - Node.js
  - Perl
  - PHP
  - And more
- There are even Docker images for languages, like:
  - Python
  - Ruby
  - Golang
  - Groovy
  - Haskell
  - Erlang
  - And more

DOCKER IMAGES FOR NEARLY ALL SERVER SOFTWARE (CONT.)

Charlie Arehart
CArehart.org
@carehart

- And even JVMs, like OpenJDK, Oracle JDK, Amazon Coretto JDK
- And even Linux distros, like Centos, Debian, Ubuntu, more
  - They all share the same Linux kernel, which Docker provides
- Finally, also packaged applications, including:
  - Monitoring solutions (ELK stack, and more)
  - Blogging software (Ghost, and more)
  - Content management systems (like Wordpress and Drupal)
  - Atlassian tools like BitBucket, Confluence and Jira
  - and more
- And best of all, you can use them with CF easily, for example…

DOCKER IMAGES FOR NEARLY ALL SERVER SOFTWARE (CONT.)

Charlie Arehart
CArehart.org
@carehart

CONSIDERING REDIS, FOR USE WITH CF FOR INSTANCE

- Redis is an open source distributed, in-memory key-value database
- CF2016 lets you use it for external session storage
  - CF2018 lets you use it (and memcached, etc.) as external distributed cache engines
- Windows users will find it challenging to find decent installer for Redis
  - No problem with Docker, which makes it almost trivial to use on any OS
- We'll see this later today, when we show integrating CF images with others

Charlie Arehart
CArehart.org
@carehart

- Let's now try to run the Adobe CF image
  - *docker run eaps-docker-coldfusion.bintray.io/cf/coldfusion*
- But you will get an error: Adobe requires acceptance of the EULA
  - We can do that using Docker's feature to pass in *environment variables*
  - We can do that using a Docker "-e" argument, and pass in an admin pw to use also
- We will also to expose CF's built-in web server port, 8500 as 8502
  - Let's also for now add also a --rm (two dashes) and a -d, also explained later
- *docker run --rm -it -p 8502:8500 -e password=123 -e acceptEULA=YES --rm -d eaps-docker-coldfusion.bintray.io/cf/coldfusion*

RUNNING YOUR FIRST CF IMAGE

Charlie Arehart
CArehart.org
@carehart

VIEWING THE RESULT OF THE CF IMAGE IN A BROWSER

- Let's now view the result
  - http://localhost:8502
- Will see display of CF built-in web server docroot, which allows directory display by default
- When CF image starts, so does that built-in web server, at port 8500 by default
  - We told Docker to expose that port to our host as 8502 (for kicks)
- Where is the web server root? It's inside the container
  - We will discuss later how to cause it to run our own files
- You could try to drill down to the CF Admin now

Charlie Arehart
CArehart.org
@carehart

- Docker uses “labels” to distinguish one version of an image from another
  - The vendor who creates the image chooses the label when building it
- We didn’t use one just now, but could
  - There is a latest-2018 and latest-2016
  - There are labels for the 5 versions of CF2018 so far, 2018.0.1 through 5, and even a .0
  - And for 2016.0.6 through the latest, 12 (from this week)

IMAGE LABELS: TRACKING DIFFERENT VERSIONS OF AN IMAGE

Charlie Arehart
CArehart.org
@carehart

"LATEST" LABEL

- There is also a "latest" label is a convention, which some love, some hate
  - Typically a vendor will post their latest current version with that label, but may not
  - Using "latest" with CF image does currently get the latest CF2018 update, 5
- What if you already have pulled/run a "latest" image in the past…
  - and the vendor puts a new version up with that label?
  - If you "run" with that label, nothing changes. Will use the one you pulled
  - If you "pull" with that label, that WILL check and in that case WOULD download new

Charlie Arehart
CArehart.org
@carehart

HOW ADOBE COLDFUSION IS LICENSED FOR USE IN CONTAINERS

- Being a commercial product, Adobe CF is subject to its EULA
- First, note that development use of CF is always free, including Docker image
- As for production use of CF on Docker, no current mention in EULA
  - See CF image docs which point to FAQ where Adobe does address this
  - CF Enterprise: can use with 8 Docker instances, per license
  - CF Standard: must license each use of a Docker instance as normal
- Not necessarily a show-stopper: most use Docker for dev, test
  - And may deploy their apps in some way other than with Docker…

Charlie Arehart
CArehart.org
@carehart

HOW ADOBE COLDFUSION IS LICENSED FOR USE IN CONTAINERS (CONT)

- Good news coming in CF2020
  - Adobe announced last month plans to improve Docker licensing
    - As well as Docker image size, startup time, and more
  - Google coldfusion docker licensing to find my blog post on this
- Finally, note that CF images deploy by default as if in Trial mode
  - And like normal will revert to Dev edition after 30 days, unless license added
  - License can be added in admin or another Docker "environment variable"
- Also implemented in "Developer Profile"
  - Can turn that off, turn on "secure profile" in admin or via another env var

Charlie Arehart
CArehart.org
@carehart

HELP FROM THE CF DOCKER IMAGE ITSELF

- Can get help from the CF Docker images themselves, using "help"
  - *docker run --rm eaps-docker-coldfusion.bintray.io/cf/coldfusion:latest-2018 help*
- Note: it's not any Docker convention for an image to offer such "help"
- Note also the start of that help shows various other "commands" CF supports…

Charlie Arehart
CArehart.org
@carehart

GETTING CF VERSION OF DOCKER IMAGE: THE "INFO" COMMAND

- Lets you see what version of CF (and CF update level) is in the image
  - Especially with a tag like 2018-latest, to see which version that is
- Let's try it:
  - *docker run --rm eaps-docker-coldfusion.bintray.io/cf/coldfusion:latest-2018 info*
- There is also a commands for CLI
  - In 2016 that allowed running CFML files from the CMD line
  - In 2018, that added a true REPL as well. Will leave you to explore these

Charlie Arehart
CArehart.org
@carehart

- The CF image “help” (and doc page) showed also several env vars for CF
  - Let’s look them over
- **password**=<Password>
- **serial**=<ColdFusion Serial Key>
  - CF serial number (aka “license key”), to enable CF to run as Standard or Enterprise
  - This and next var were not available until Docker image labels 2016.0.11 and 2018.0.2
- **previousSerial**=<ColdFusion Previous Serial Key (Upgrade)>
  - Serial number for your previous version, if one above is "upgrade" license, which requires you to also specify previous version serial number

BASIC CF CONFIGURATION ENV VARS

Charlie Arehart
CArehart.org
@carehart

- These (like the last) are things we are asked during install of CF, but no installer
- **enableSecureProfile**=<true/false(default)>
- **configureExternalAddons**=<true/false(default)>
- **configureExternalSessions**=<true/false(default)>
  - More on these two and related vars, coming up very shortly
- **language**=<ja/en (Default: en)>

ENV VARS TO CONFIGURE CF SETTINGS

Charlie Arehart
CArehart.org
@carehart

ENV VARS TO CONFIGURE OTHER CF SETTINGS

- **setupScript**=<CFM page to be invoked on startup. Must be present in the webroot, /app>
  - Can run any CFML at all, such as Admin API
  - Will see this and other ways to auto-configure a CF Docker container, later
- **setupScriptDelete**=<true/false(default) Auto delete setupScript post execution>

- What about env var for editing JVM config?
  - Sadly there is none (Tomcat's image has one, CF's does not)
  - Hope this is addressed in CF2020

Charlie Arehart
CArehart.org
@carehart

- Add-on service is optional during normal install
  - Runs CF's Solr (test index) engine for use with CFINDEX/CFSEARCH, etc.
    - And CF's PDFG (html-generating webkit implementation), for CFHTMLTOPDF
- If enabled in Docker with *configureExternalAddons=true*, can also set:
  - **addonsHost**=<Addon Container Host (Default: localhost)>
  - **addonsPort**=<Addon Container Port (Default: 8989)>
  - **addonsUsername**=<Solr username (Default: admin)>
  - **addonsPassword**=<Solr password (Default: admin)>
  - **addonsPDFServiceName**=<PDF Service Name (Default: addonsContainer)>
  - **addonsPDFSSL**=<true/false(default)>

ENV VARS TO CONFIGURE THE CF ADD-ONS SERVICE

Charlie Arehart
CArehart.org
@carehart

- CF2016 added option to store CF session variables in a Redis server
  - can be setup as another image or as available externally to Docker
- If enabled in CF Docker image using *configureExternalSessions=true*, can set:
  - **externalSessionsHost**=<Redis Host (Default:localhost)>
  - **externalSessionsPort**=<Redis Port (Default:6379)>
  - **externalSessionsPassword**=<Redis Password (Default:Empty)>

ENV VARS TO CONFIGURE THE CF EXTERNAL SESSIONS FEATURE

Charlie Arehart
CArehart.org
@carehart

CONFIGURING CONTAINER VIA ENV FILE

- We've been passing env vars on **docker run**
  - Mentioned previously that docker supports passing them in as a file, a .env file
  - Plain text file, that lists env var=value pairs on separate lines
- Example: c:/coldfusion.env
  - acceptEULA=YES
  - password=123
- Docker run offers ***--env-file*** arg
  - *docker run -p 8500:8500* --env-file c:/coldfusion.env *-d --rm eaps-docker-coldfusion.bintray.io/cf/coldfusion:latest-2018*

Charlie Arehart
CArehart.org
@carehart

DIFFERENT VARIANTS OF CF DOCKER IMAGES

- Before we move on to discussing the CF Docker images further, take note
- Ortus also offers CF Docker images (more in a moment)
  - Anyone here using those? Expected us to be using those instead?
  - There are many resources on using CF and Docker that way, not with CF images
  - Much of what we discuss today applies to Docker in general, and thus those also
- Also, may find other "CF" images on Dockerhub: from private folks, self-built
- Focus here is obviously on Adobe CF images
  - But let's look briefly at the Ortus image

Charlie Arehart
CArehart.org
@carehart

- See https://hub.docker.com/r/ortussolutions/commandbox
  - See available env vars, options
  - Available for CF2018, 2016, and 11, as well as Lucee
  - Quick example: ***docker run -p 8080:8080 ortussolutions/commandbox:adobe2018***
    - It creates CF Admin password of its own, if none passed in via config file
    - Visit via normal CF Admin URL (their image uses nginx, does not expose dir listing)
- Other advantages
  - Different env var options
  - Additional config options: server.json, cfconfigfile, box.json
  - Option for no admin (headless)
  - To name a few

ORTUS COMMANDBOX CF IMAGE

Charlie Arehart
CArehart.org
@carehart

ORTUS COMMANDBOX CF IMAGE (CONT)

- Some Adobe CF image advantages/differences of note
  - Configure via car
  - Config script option, option to delete that after start
  - Env vars for serial number, enabling sec profile, enabling/config of redis sessions, enable/config of add-ons, etc
  - Availability of those other CF images: addons, pmt, api mgr
- What about differences in startup times, sizes?
  - Once you have the Ortus Docker image obtained for CF, it's about the same for both

Charlie Arehart
CArehart.org
@carehart

- At this point we now have 3 running CF containers, at 8500, 8501, 8502
  - How would you know? I'll show you, and how to manage them
- But note how easily we have 2 CF "instances"
  - You couldn't run the CF installer twice on one machine
- Yes, CF Enterprise/Trial/Dev supports multiple instances
  - But this was again faster, easier—and not permanent
- Now's a good time to talk about commands to manage docker

MID-WAY CHECKPOINT

Charlie Arehart
CArehart.org
@carehart

LISTING CONTAINERS

- Listing containers: ***docker ps*** (try it)
  - About container identifiers: container id, image name, container name
    - Name chosen randomly, unless you set, as we will see how later
  - We will use id or name to refer to containers in later commands
- Listing stopped containers: ***docker ps –a*** (try it)
  - Notice all the stopped containers we have
    - Would have had more but recall we used "--rm" arg once. Removes on stop
  - Will learn how to remove stopped ones soon
- More recent versions of docker support also ***docker container ls***
  - Accepts same args, produces same output

Charlie Arehart
CArehart.org
@carehart

GETTING HELP FOR DOCKER COMMANDS

- Good time to take a detour to show how to get help for docker commands
  - Usually can add --help to the command, so ***docker ps --help***
- Can also get list of all Docker commands, with ***docker --help***, or just ***docker***
  - Can be overwhelming. Really, only a handful of commands are used often
- Better still, see excellent online Docker help, like
  - https://docs.docker.com/engine/reference/commandline/ps/

Charlie Arehart
CArehart.org
@carehart

- Recall we found our CF and other containers running. What if we're done?
- Can stop a container with ***docker stop <container>***
  - Can use container name or container id, obtained with **docker ps**
  - Let's stop our first CF instance
- Can kill a container with ***docker kill <container>***
  - Kill is more abrupt than stop, as stop can wait up to 10 seconds for graceful shutdown
  - Try killing the other CF instance (difference compared to **stop** will vary)
- It's pretty annoying typing those long container names or ids…

STOPPING CONTAINERS

Charlie Arehart
CArehart.org
@carehart

- Wonderful shortcut: just need enough chars for id (not name) to uniquely identify
  - So if only one container had an id starting with c1, can use just ***docker stop c1***
  - Try it for one of the CF containers we left running. Sweet!
- This applies to all commands that act on containers

PRO TIP 1: NEED ONLY TYPE PART OF ID

Charlie Arehart
CArehart.org
@carehart

YOU CAN START A STOPPED CONTAINER; RESTART A RUNNING ONE

- Use ***docker start <container>***
  - Will simply start it again, unless something would prevent that
- Can also restart a running container: ***docker restart <container>***
  - As expected, will stop and then start it
- And the content within it remains as it was before stop
  - Let's restart a CF container, after first setting a setting in the CF Admin, like req timeout

Charlie Arehart
CArehart.org
@carehart

REMOVING STOPPED CONTAINERS

- What if you really no longer need a stopped container?
  - Recall that ***ps –a*** shows those stopped but still existing
- Can remove stopped containers with ***docker rm <container>***
  - Again using name or containerid (or a unique starting portion of that id)
  - Let's try it
  - "modern" name is ***docker container rm <container>***
- Want to remove all stopped containers? ***docker container prune***
  - See docs for more on -f arg, for filtering what containers are pruned
- Recall also that the --rm argument, on docker run, will remove container on stop
- Note that stopping docker or your host will stop, but not remove, containers

Charlie Arehart
CArehart.org
@carehart

- You will often find you have several stopped that you want to remove
  - But you may not want to use **container prune** (of all, or figuring a filter to use)
- Check this out: you can name multiple ids (or enough of id to be unique)
  - So could do something like ***docker rm c1 23 5f***
  - Let's create a few hello-world images to test that!
  - Works also on **stop** command, and some others

PRO TIP 2: CAN LIST MULTIPLE IDS AT ONCE

Charlie Arehart
CArehart.org
@carehart

WHAT HAPPENS TO INFO "IN" CONTAINER WHEN REMOVED

- We've seen that a stopped container can be restarted, and still shows info "inside it"
  - But what happens when you remove a container?
  - In that case, the info inside the container (like the container itself) is "gone"
- This argues for containers to be stateless: to not have changes made, where their loss would be a problem
  - An analogy of cattle vs pets is often used for this
- But what if you do need to change (and save) data in a container?
  - There's a feature for that, coming up soon

Charlie Arehart
CArehart.org
@carehart

- Focus to now has been managing containers, now let's discuss images
  - Many commands have parallels
- To list all Docker images you have: ***docker image ls***
  - or just ***docker images*** (note that ***docker containers*** does not exist, for now)
- And as with docker ps, see --help or online help for available args
- To remove an image: ***docker rmi <image>*** (image name/label or id)
  - Removing image does indeed remove the image from your docker host
  - Does not work with only partial imageid, like docker ps and rm
  - More modern variant: ***docker image rm <image>***
- ***docker image prune***
  - Can removes all unused and/or "dangling" unused images. See docs for more

MANAGING IMAGES

Charlie Arehart
CArehart.org
@carehart

AVAILABLE UI'S TO MANAGE DOCKER CONTAINERS, IMAGES

- While cmd line is king with Docker, not everyone loves it
  - Lots to remember, though in time it becomes natural—and remember pro tips!
- But there are options for UI-based docker mgt of containers, images
- Most popular may be portainer
  - Try this:
  - ***docker run -d -p 9000:9000 -v portainer_data:/data portainer/portainer***
  - Then visit localhost:9000 to see its ui (choose a password on first visit)
- If you use VisualStudio Code, see its nice Docker plugin

Charlie Arehart
CArehart.org
@carehart

- Volumes are an important feature in Docker
  - Allows you to "poke a hole" in the isolation of container files from host files
  - Using them allows things like run your own code in a web server or in CF
    - Or storing data modified by container across container removals (databases, logs, uploads)
- Another topic worth of perhaps a day, but let's show a basic use
  - ***docker run*** has a –v arg, can be used to map a folder on host to one in container
  - Recall that the CF image "help" showed an available /app folder
    - This is where that CF built-in web server looks for code
  - So we could run a CF image telling it to map a folder on our host to that…

CONFIGURING VOLUMES

Charlie Arehart
CArehart.org
@carehart

CONFIGURING VOLUMES

- Let's say we had webcode at c:\inetpub\wwwroot
  - We could point to/"mount" that as ***–v c:/inetpub/wwwroot:/app***
  - Windows users: note need to use / instead of \
  - If we do that with CF docker image, we will run code in that folder
  - And still have available CFIDE folder, for use by admin
- See much more in docker docs, resources about other uses of volumes
  - Concept of named volumes
    - https://docs.docker.com/storage/volumes/
  - Technically, -v is a less powerful/less flexible "bind mount"
    - https://docs.docker.com/storage/bind-mounts/
  - Also, one container can mount volumes from another, with ***--volumes-from***

Charlie Arehart
CArehart.org
@carehart

- All discussion to this point has been about configuring container at run time
- Docker also lets us create our ***own*** image
  - Can be useful in many cases
  - Involves creating a "dockerfile", in which you specify directives to build it
  - Is simple in concept and execution
    - In interest of time, I'll leave that as an exercise (see example in Adobe CF docs page)

BUILDING IMAGES

Charlie Arehart
CArehart.org
@carehart

SAVING CONFIGURED STATE: DOCKER COMMIT

- Finally, what if you wanted to save the state of an image, once configured?
  - Can use ***docker commit***
  - Will create new image, based on one named, with new name
- Beware, it will be as large as or could be bigger than the original
  - Recall ways to manage images
  - Cold then push that modified image to a registry with ***docker push***
  - Do beware not to push to public registry any image with sensitive config data

Charlie Arehart
CArehart.org
@carehart

- First option that CF provides is to use the CF "CAR" export/import feature
  - Has been in CF Admin in Standard/Enterprise (and Trial/Dev) since CF11
  - Prior to CF11, was only Ent/Trial/Dev
- If you create such a .car file (from some CF admin)
  - Can then place that .car file in a folder mounted as volume to image's /data directory
  - Automatically imported during startup. Demo?

CF IMAGE'S CAR IMPORT FEATURE

Charlie Arehart
CArehart.org
@carehart

- The second option is that env var for ***setupScript***
  - This would name a cfm file, expected to be in image's /app folder (or volume mounted to it)
- In that template, can run any CFML, but most likely use is for CF Admin api
  - This is a powerful set of CFCs (and their methods) that can support doing pretty much any CF admin task, programmatically
  - Added in CF7, not used widely in my experience
  - See my blog post: https://www.carehart.org/blog/client/index.cfm/2018/1/3/great_start_on_adminapi_docs
- Remember also that setupScriptDelete env var
  - If you may desire to delete the named file from container once it's started

CF IMAGE'S "SETUP SCRIPT" FEATURE

Charlie Arehart
CArehart.org
@carehart

CFCONFIG

- Another useful option for configuring admin settings is ***cfconfig***, from Ortus
  - Including editing, extracting, comparing and doing other options
- Originally a commandbox module, can be used with more than that
  - Is built-into Ortus Commandbox docker image, but not CF's, though you could add it
  - See https://cfconfig.ortusbooks.com

Charlie Arehart
CArehart.org
@carehart

MANAGING MULTIPLE IMAGES

Charlie Arehart
CArehart.org
@carehart

- Docker compose is a tool to facilitate starting/managing groups of images
  - Mostly a development tool (a "first step" toward orchestration)
- Organizing several configuration steps into one file
  - A yml (yaml) file. Just plain text, quite easy to understand
  - Docker compose (and docs) define the format, simple examples will explain
  - (CF Docker help page has some rather obtuse examples)
- Especially useful for organizing multiple related images, as we shall see
- We use ***docker-compose up*** command, optionally naming yml file

ABOUT DOCKER COMPOSE

Charlie Arehart
CArehart.org
@carehart

DOCKER-COMPOSE OF JUST THE CF IMAGE

- In the vein of our walk-before-we-run motif, let's setup the yml for running just CF
  - Create a folder, to hold these, such as c:\docker-compose
  - In that folder, create another called c:\docker-compose\cf-alone
  - In that folder, create or copy earlier coldfusion.env file, with:

    acceptEULA=YES

    password=123
  - And create there a docker-compose.yml with text on following page…

Charlie Arehart
CArehart.org
@carehart

```
version: "3"
services:
    cf:
        container_name: cf
        image: eaps-docker-coldfusion.bintray.io/cf/coldfusion:latest-2018

        ports:
        - "8500:8500"

        env_file:
        - coldfusion.env
```

DOCKER-COMPOSE OF JUST THE CF IMAGE (CONT)

Charlie Arehart
CArehart.org
@carehart

DOCKER-COMPOSE OF JUST THE CF IMAGE (CONT)

- Note that indentation must be present and consistent (tabs or spaces)
- Now go to command line, to that cf-alone directory
  - Issue ***docker-compose up***
- If all goes well, display of logs will show CF coming up
  - When complete, should be able to visit localhost:8500 as before
- Just like with **docker run**, without -d, we see logs from compose svc(s)
  - Unlike docker run, if we use ctrl-c to get back to cmd line, stops svc(s) in compose
  - Could use –d when starting it, as in ***docker-compose up -d***
  - Can also open another command prompt and work from there also

Charlie Arehart
CArehart.org
@carehart

- Just like with docker mgt of container, we can manage compose
  - docker-compose ps
  - docker-compose exec
  - docker-compose logs
  - docker-compose kill, and more
  - Run docker-compose (no args) for commands, and --help to any cmd
- Can do ***docker-compose down***, to bring down what was brought up
  - Do need to be in the same directory as the docker-compose.yml file
- Do that also to remove svc containers, if you ctrl-c on up (while showing logs)

MANAGING DOCKER-COMPOSE

Charlie Arehart
CArehart.org
@carehart

DOCKER COMPOSE FOR OTHER INTEGRATING OTHER CF IMAGES

- Let's look at
  - CF and PMT
  - CF and Redis for CF images

Charlie Arehart
CArehart.org
@carehart

TROUBLESHOOTING DOCKER

Charlie Arehart
CArehart.org
@carehart

- Most images will create some form of logs, very useful for debugging
  - If we just run (without -d, for "daemon mode"), the container will show logs on screen
  - Let's try starting a CF image without that
  - *docker run -p 8500:8500 -e acceptEULA=YES --rm eaps-docker-coldfusion.bintray.io/cf/coldfusion:latest-2018*
  - If we wanted to get back to cmd line, can use ctrl-c
- Can also access logs for a container using ***docker logs <container>***
- Note that this is only whatever logs are written to "stdout"
  - In CF, this is primarily what's in coldfusion-out.log
  - There are solutions to get more logged, beyond scope of this workshop

DOCKER LOGS

Charlie Arehart
CArehart.org
@carehart

- Docker offers a command to execute commands inside of container
  - Can be very helpful for debugging container issues
  - Use ***docker exec <container> <cmd>***
  - For example, can issue a Linux LS (on Linux image) as ***docker exec <container> ls***
  - For Linux images (like CF's) which support bash shell, can get to it within container
    - ***docker exec <container> bash***
    - Then can use any available Linux commands. Some things you expect may not be there
    - Can exit with "exit"

EXECUTING CMDLINE INSIDE IMAGE

Charlie Arehart
CArehart.org
@carehart

- Nifty tool to see resource use of running containers
  - Simply **Docker stats**
  - Note how it stays on-screen, refreshing
  - Cancel with ctrl-c, see also --no-stream arg to get just a single line output
- Really much more we could talk about regarding docker troubleshooting
  - Again, could be its own day. Just whetting your whistle here

DOCKER STATS

Charlie Arehart
CArehart.org
@carehart

- With Docker Desktop, for Windows, go to tray, right-click on Docker icon
  - Go to Advanced, see sliders there to control
  - Set memory especially to size suited to your available ram
    - And what CF or other images you're running, their memory (heap) size, etc.

MAY NEED TO INCREASE MEM, DISK, CPU ALLOCATED TO DOCKER

Charlie Arehart
CArehart.org
@carehart

- You can monitor docker images from tools outside the image (such as via http)
- You can also implement monitoring WITHIN the image
  - There are many generic monitoring/APM tools
  - To see how to monitor CF image with FusionReactor, see
    - https://github.com/intergral/fusionreactor-docker/tree/master/coldfusion
- And of course we saw previously the available CF PMT image

MONITORING DOCKER IMAGES

Charlie Arehart
CArehart.org
@carehart

WRAPPING UP

Charlie Arehart
CArehart.org
@carehart

- Once you have yml for compose, it's barely another step to use docker swarm
  - Which can add much more capability to manage instances, across machines
  - It's included with Docker Desktop, so easy to try out. Also at PWD
  - Matt Clemente had a session on that yesterday, and nice blog posts
- Kubernetes is the next evolution in orchestration
  - It too is offered in Docker Desktop
- And then you can deploy your images and orchestration on cloud providers like AWS, Azure, Google Cloud, Digital Ocean

MOVING TO ORCHESTRATION

Charlie Arehart
CArehart.org
@carehart

- We're about out of time. I just wanted to at least plant a seed for you
  - If you are already doing CI/CD, you should explore Docker for what it brings to that
  - More capabilities for automated testing, deployment, app upgrades, etc.
- Google searching on the topic will find ample discussions

DEVELOPMENT/TESTING AND CI/CD

Charlie Arehart
CArehart.org
@carehart

- Phew, so that was our whirlwind tour of Docker, and CF Docker images
  - Hope you better appreciate value of running Docker and especially using it with CF
  - and how using it with still other software can make your job easier
- Let's end with "10 Reasons Developers Love Docker" (from Docker) …

WHY DEVELOPERS LOVE DOCKER

Charlie Arehart
CArehart.org
@carehart

WHY DEVELOPERS LOVE DOCKER (CONT)

- It works on everyone's machine
- Takes the pain out of CI/CD
- Boosts your career
- Makes cool tech accessible
- Raises productivity
- Standardize Development + Deployments
- Makes cloud migration easy
- Application upgrades are a lot easier
- And, if an app breaks, it's easy to fix
- It's easy to try out new apps

Charlie Arehart
CArehart.org
@carehart

LEARNING MORE ON YOUR OWN

- There are some topics we've only barely touched on
  - And some we've not even covered (that some may think should have been)
- See resources section in appendix for many useful resources
  - Many geared to getting started
  - Some organized as courses of their own (many free or cheap)
- I will offer a blog post soon pointing to many that have helped me

Charlie Arehart
CArehart.org
@carehart

GOING FORTH

- Try running again the various demos, to reinforce points
- Raise any issues on the Adobe bug tracker (tracker.adobe.com)
  - Once you choose CF, it has "components" for "containers – docker …"
- I plan a 30-days to "Getting Started with Docker and the CF Images"
- Reach out to me for question on the materials, specific topics covered
  - charlie@carehart.org, or on most social media as carehart
- One last thing: trying to start the CFMeetup again, see coldfusionmeetup.com
- Any questions now? (if we have time)

Charlie Arehart
CArehart.org
@carehart