Docker cheat sheet pdf

$ docker container run-it-p 80:80 nginx $ docker container run -d -p 80:80 nginx $ docker container run -d-p -p-p 80:80 nginx - docker container run -d -d -p 80:80 --name nginx-serverinx-searched image called nginx in cache images If not found in cache It looks at the default repo image on Dockerhub pulled it down (the latest version) stored in the cache of images Started it in a new container We indicated to take port 80- on the host and forward to port 80 on the container We could make a $ docker container run --publish 8000:80 --disconnect nginx to use port 8000 We can specify, as nginx:1.09 Or List of all containers (even if not working) Stop the container $ docker container stop ID Stop all running containers $ docker stop $$(docker ps-aq) Remove the container (can't remove the running containers must stop first) $ docker container rm (ID) To remove the running container use force (-f) $ docker container rm-f (ID) Remove several containers $ docker container rm (ID) (ID) Remove all containers $ docker rm $rm $rm (docker ps-aq) Get logs (Use name or ID) $ docker containers magazines NAME List processes, Working in a container $ docker container top NAME TIP: About CONTAINERS Docker containers are often compared to virtual machines, but they are actually just processes running on the host OS. In Windows/Mac Docker works in a mini-VM to see the processes that you need to connect directly to this. On Linux, however, you can run ps aux and see the processes directly IMAGE COMMANDS $ Docker image rm IMAGE $ docker rmi $(docker image -a-q) Images of the application bianaries and dependencies with meta-data on image data and how to run the image Images are not a complete OS. No core, core modules (drivers) Host provides the core, big difference between VM NGINX: $ docker container run -d -p 80:80 --name nginx nginx (-p 80:80 is optional as it works at 80 by default) APACHE: $ Docker container run -d -d -p 8080:80 - Apache name httpd MONGODB: $ docker container run-d-p:27017 --name mongo MYS'L: $ docker container run -d-p 3306:3306 --name mysql --env MYSQL_ROOT_PASSWORD '123456 mysql CONTAINER INFO View information on container $ docker container check (NAME) Specific property (-format) $ docker container check -- format'. NetworkSettings.IPAddress (NAME) Performance Stats (cpu, mem, network, drive, etc.) $ docker container stats (NAME) ACCESSING CONTAINERS Create a new container nginx and bash in a $ docker container run - it's the name NAME nginx bash i interactive Keep STDIN open if not attached t . Use winpty $winpty docker container run-it-name NAME nginx bash run/ Ubuntu container $ docker container run - it's - the name ubuntu ubuntu (no bash, because ubuntu uses bash by default) You can also do it when you get out of the container not to stay using the -rm flag $ docker container run - - Access to an already created container, start with -ai $ docker container start -ai ubuntu Using exec to edit config, etc. $ docker container exec-it mysql bash Alpine is a very small Linux distribution good for a docker $ docker container run-it's alpine sh (use sh, because it doesn't include bash) (Alpine uses apk for its package manager - can set the bash, NETWORK_NAME if you want) NETWORKING $ docker container port NAME $ Docker Network check NETWORK_NAME name (NAME) --network (NETWORK_NAME) nginx $ docker network connect (NETWORK_NAME) CONTAINER_NAME $ Docker network disconnect (NETWORK_NAME) CONTAINER_NAME the network will disconnect (NETWORK_NAME) that each image has a retag tag of the existing image $ docker tag nginx btraversy/nginx Download dockerhub $ docker image push bradtraversy/nginx If denied, Add a tag to the new image $ docker image tag bradtraversy/ nginx bradtraversy/nginx : DockerFILE PARTS FROM testing - OS is used. Common Alpine, debian, ubuntu ENV - Environment variable RUN - Running commands/shell scripts, etc. EXPOSE - Ports expose CMD - Final team launch at the launch of a new container from the workdIR image - Work directory kits (also can use 'RUN CD/some /way') COPY - Copies of files from host to container Build image from dockerfile If you change one line and re-run, this line and all after will not cache Hold things that change most to the bottom of Dockerfile EXTENDING DOCKERFILE from nginx: the latter - Expands nginx so that's it, What is included in this image is included here WORKDIR /usr/share/nginx/html COPY index.html index.html $docker image build -t nginx-site $usker container run -p 80:80 --rm nginx-site $ docker image tag nginx-site: last btraversy/nginx-site: last $us-80-image-bradtraver sy/nginx-site$docker image check mysql $ docker container run-d--name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD 'True mysql $ docker container check mysql You'll also see the volume under the mounts container gets its own uniqe location on the host to store this data Source: xxx where he lives on the host There is no way to distinguish the volumes from each other for example, with 2 mysql containers, so we used the named volume Named Volumes (Add -v Team) (the name here is mysql-db, which can be anything) $ docker container run -d-d --name mysql -e MYSQL_ALLOW_EMPTY_PASSWORD 'Truth -v mysql-db:/var/lib/mysql mysql Inspect new named volume docker volume check mysql-db BIND MOUNTS can't be used in specified time time to run run -v также) ... run-v /Users/brad/stuff:/path/container (mac/linux) ... run-v //c/Users/brad/stuff:/path/container //c/Users/brad/stuff:/path/container Tip: Instead of entering a local path, use $(pwd)/path/container for the work directory If you are in the Run user folder and will be able to edit the index.html file (local dir must have Dockerfile and index.html) $ docker container run -r 80:80 -v $(pwd):/usr/share/nginx/html nginx Jump into the container and check the $ docker container exec-it nginx bash $ CD/usr/share/nginx/HTML$ls-al You could create a file in a container, and it will exiost on the host, as well as THE COMIC GRID Setting of the relationship between the containers Save our settings docker container run in easy read file 2 Parts: YAML file (docker.com.yml) - CLI tool (docker-composition) containers network volumes version: '2' - just like the docker run -p 80:4000 -v $(v $(pwd): / site bretfisher : Jekyll: image: bretfisher/jekyll-serve volume: --:/ Ports site: - '80:4000' Git Repo: Docker to build 'options'. -t app/container_name - name --build-arg APP_HOME'$APP-HOME - Install build time variables Create an image from Dockerfile. Docker run docker run options IMAGE see Docker Create for options Example $ Docker run-it debian:buster/bin/bash You run the team in the image. Manage docker containers to create dockers to create options IMAGE -a,-attach - attach-thick/bug -i,-interactive - attach stdin (interactive) -t,---tty --tty --name-name -name -p, --publish 5000:5000 - port map (host:container) --expose 5432 --expose-port-publish-all-publish-all-post all ports --link container:-alias -linking-v,-volume 'pw d':/app) mount (absolute paths are needed) -e,-env NAME'hello - env vars Example $docker create --name app_redis_1 --expose 6379 and redis:3.0.2 Create a container from the image. Docker Exec Docker Exec (options) CONTAINER COMMAND -d,--detach -- run in the background -i,-interactive-stdin-t,-t-tty-tty-interactive example of $ Docker Exec app_web_1 tail logs/development.log$-dot-exec-t-i app_web_1 rails with running teams in a container. The docker start the docker start the OPTIONS CONTAINER -a,-attach -- attach-fat/bug-i-interactive - attach Stdin docker to stop container start/stop container options. Docker PS $Docker PS$Docker PS -A $Docker kill $ID to manage containers using PS/kill. Docker magazines $ docker magazines $ID $ Dockers magazines $ID 2'gt;'1 less than $ docker magazines -f $ID - Follow the magazine's release Look that is logged in the container. Image copyright Docker Image caption REPOSITORY TAG ID ubuntu 12.10 b750fe78269d me/myapp past 7b2431a8d968 $968 to show interim image control. Docker rmi docker rmi b750fe78269d Removes Cleaning clean all docker prunes system clears dangling images, containers, volumes and nets (i.e., not related to the container) docker prunes system- besides removing any stopped containers and all unused images (not just dangling dangling Containers - Stop all working docker containers stop $(docker ps-a-q) - Remove stopped containers of docker container prunes image of the prunes image and all images of the volume of docker prunes Remove all volumes Also see start (docker.io) Full Docker CLI Control container control container CLIs Checking container interaction with container image of the container_name image management team The Docker run docker to run options IMAGE see the docker create for the options you run the team in the image. Manage docker containers to create docker create options IMAGE -a,-attach - attach-thick/bug-i,-interactive - attach stdin (interactive) -t,--tty --tty --name NAME - name your image -p, --publish 5000:5000 - port map --expose 5432 - expose port-publish-all-publish all ports-link container: alias - link-v,--volume 'pwd':/app - mount (absolute paths necessary) -e, --env NAME'hello - env vars Example $ docker to create --name app_redis_1 --expose 6379 and redis:3.0.2 Create a container from the image. Docker Exec Docker Exec (options) CONTAINER COMMAND -d,--detach -- run in the background -i,-interactive-stdin-t,-t-tty-tty-interactive example of $ Docker Exec app_web_1 tail logs/development.log$-dot-exec-t-i app_web_1 rails with running teams in a container. The docker start the docker start the OPTIONS CONTAINER -a,-attach -- attach-fat/bug-i-interactive - attach Stdin docker to stop container start/stop container options. Docker PS $Docker PS$Docker PS -A $Docker kill $ID to manage containers using PS/kill. Image copyright Docker Image caption REPOSITORY TAG ID ubuntu 12.10 b750fe78269d me/myapp past 7b2431a8d968 $968 to show interim image control. Docker rmi Removes images. Also see (docker.io) Dockerfile ENV APP_HOME /myapp RUN mkdir $APPHOME VOLUME /data - Specification for the point of mount ADD file.xyz/file.xyz COPY --chown-user:group host_file.xyz/path/container_file.. xyz ONBUILD RUN bundle set when used with another FILE EXPOSE 5900 CMD bundle, exec, rails, server - ENTRYPOINT execution, param1, param2 - COMANDA ENTRYPOINT param1 param2 Sets up a container that will work as you performed. In this case, the shell processing will be used to replace the shell variables and will ignore any arguments of the command line CMD or docker. LABEL com.example.vendorACME Incorporated LABEL com.example.label-with-valuefoo LABEL This text illustrates that label values can cover multiple lines. See also and docker-compose.yml version: '2' services: web: build: . assemblage from the Context Dockerfile:./Path dockerfile: Dockerfile Ports: - - объемы: -:/ код переделис: изображение: redis докер-составить начать докер-составить стоп докер-составить паузу докер-сочинять непаузи докер-составить докер-составить докер-составить вниз веб: # build from custom Dockerfile build: context: ./dir dockerfile: Dockerfile.dev # build from image image: ubuntu image: ubuntu:14.04 image: tutum/influxdb image: example-registry:4000/postgresql image: a4bc65fd ports: - 3000 - 8000:80 # guest:host # expose ports to linked services (not to host) expose: [3000] # command to execute command: bundle exec thin -p 3000 command: [bundle, exec, thin, -p, 3000] # override the entrypoint entrypoint: /app/start.sh entrypoint: [php, -d, vendor/bin/phpunit] # environment vars environment: RACK_ENV: development environment: - RACK_ENV=development # environment vars from file env_file: .env env_file: [.env, .development.env] # makes the 'db' service available as the hostname 'database' # (implies depends_on) links: - db:database - redis # make sure 'db' is alive before starting depends_on : - db # make this service extend another extends : файл: common.yml - дополнительный сервис: объемы веб-приложений: - /var/lib/mysql - ./_data:/var/lib/mysql услуги: веб: этикетки: com.example.description: Бухгалтерский веб-приложение услуги: веб: dns: 8.8.8.8 dns: - 8 .8.8.8 - 8.8.4.4 услуги: web: устройства: - услуги /dev/ttyUSB0:/dev/ttyUSB0: web: external_links: - redis_1 - project_db_1:mysql услуги: web: extra_hosts: - somehost:192.168.1.100 Чтобы просмотреть список всех услуг, бегущих в рое, чтобы увидеть все службы docker стек услуг stack_name чтобы увидеть все журналы услуг docker службы журналы stack_name service_name Чтобы масштабировать услуги быстро через квалифицированный узел докеров масштаба службы stack_name_service_name чернослив неиспользованные (dangling) изображения Для удаления всех изображений, которые не используются контейнеры, добавить - чтобы Purne всю систему, чтобы оставить рой, чтобы удалить рой (удаляет все данные об объеме и информацию о базе данных) доккер стек rm stack_name Чтобы убить всех бегущих контейнеров докер убить $ (docekr PS -q) Автор - Sangam biradar - Лидер сообщества Docker docker cheat sheet github. docker cheat sheet pdf 2020. docker cheat sheet edureka. docker cheat sheet for spring developers. docker cheat sheet with examples. docker cheat sheet intellipaat. docker cheat sheet pdf 2018. docker cheat sheet atlassian