
Tutoriel Docker

Release 2018-11-08T12:52:56.864807+01:00

2018-11-08T12:52:56.864807+01:00

Contents

1	Introduction à Docker	3
1.1	Pourquoi utiliser docker ?	4
1.1.1	Transformation de la DSI des entreprises	4
1.1.2	Pour donner davantage d'autonomie aux développeurs	5
1.1.3	Faire évoluer son système d'information	5
1.1.4	Pour que ça fonctionne aussi sur une autre machine	6
1.1.5	Livre blanc Ubuntu	6
1.2	Définitions concernant l'agilité et le mouvement Devops	6
1.2.1	Définition de Devops p.34 Programmez! p.214 janvier 2018	6
1.2.2	Définition 2, Le Devops pour répondre à l'appel de l'innovation 2018-01-04	7
1.2.3	Définition 3, extrait p.53 MISC N95, Janvier/février, 2018, “ Ne pas prévoir, c'est déjà gémir ”	7
1.2.3.1	Citations	8
1.2.3.1.1	Ne pas prévoir, c'est déjà gémir	8
1.2.3.1.2	La vie, c'est comme une bicyclette, il faut avancer pour ne pas perdre l'équilibre	8
1.2.4	Devops, intégration et déploiement continus, pourquoi est-ce capital et comment y aller ?	8
1.2.5	Agilité et Devops : Extrait p. 35 de [Programmez!] , N°214, janvier 2018	9
1.2.6	What is a DevOps Engineer ?	9
1.3	Définitions concernant Docker	10
1.3.1	Définition de Docker sur Wikipedia en français	10
1.3.2	Docker est “agile”	10
1.3.3	Docker est portable	10
1.3.4	Docker est sécurisé	14
1.3.5	Les conteneurs Docker sont plus légers et rapides que les machines virtuelles	14
1.3.5.1	Containers	14
1.3.5.2	Virtual machines (VMs)	14
1.3.5.3	Docker can run your applications in production at native speed	14
1.4	Dossier Docker dans le dossier MISC N°95 de janvier/février 2018	15
2	Installation de docker	17
3	Qui utilise Docker en production ?	19
3.1	Historique	19
3.1.1	Janvier 2018	19
3.2	Paypal	20
3.2.1	Challenges	20
3.2.2	Solution	20

4 docker engine CE (Community Edition)	23
4.1 docker engine versions	23
4.1.1 docker engine versions	23
4.1.1.1 Future	24
4.1.1.2 Versions	24
4.1.1.2.1 18.09-ce (2018-11-08)	24
4.1.1.2.2 18.06.1-ce (2018-08-21)	24
4.1.1.2.3 18.06.0-ce (2018-07-18)	24
4.1.1.2.4 18.03.1-ce (2018-04-26, 9ee9f40)	24
4.1.1.2.5 17.12.1-ce (2018-02-27)	24
4.1.1.2.6 17.06.0-ce (2017-06-23, 02c1d87)	24
5 Docker compose	25
5.1 Concepts clés	25
5.1.1 Other definition	26
5.2 Other links	26
5.2.1 heroku	26
5.3 docker-compose commands	26
5.3.1 docker-compose commands	26
5.3.1.1 docker-compose help	27
5.3.1.2 docker-compose build	28
5.3.1.2.1 docker-compose -f docker-compose.yml build django	28
5.3.1.3 docker-compose ps	28
5.3.1.4 docker-compose up	29
5.4 docker-compose for production	30
5.4.1 Docker compose in production	30
5.4.1.1 Articles	30
5.4.1.1.1 Simple continuous deployment with docker compose, docker machine and Gitlab CI (2017-06-26)	30
5.5 docker-compose TIPS	31
5.5.1 docker-compose tips	31
5.5.1.1 docker-compose tips 2018	31
5.5.1.1.1 3 Docker Compose features for improving team development workflow	31
5.6 docker-compose versions	33
5.6.1 docker-compose versions	33
5.6.1.1 Versions	34
5.6.1.1.1 docker-compose 1.22 (2018-07-18)	34
5.6.1.1.2 docker-compose 1.21 (2018-04-09)	34
5.7 Exemples	34
5.7.1 Quickstart: Compose and Django	34
5.7.1.1 Overview of Docker Compose	35
5.7.1.2 Introduction	36
5.7.1.3 Define the project components	36
5.7.1.3.1 mkdir django_app	36
5.7.1.3.2 Create a Dockerfile	36
5.7.1.3.3 Create a requirements.txt in your project directory	37
5.7.1.3.4 Create a file called docker-compose.yml in your project directory	37
5.7.1.4 Create a Django project	39
5.7.1.4.1 cd django_app	39
5.7.1.4.2 docker-compose run web django-admin.py startproject composeexample	39
5.7.1.5 Connect the database	41
5.7.1.5.1 Edit the composeexample/settings.py file	41
5.7.1.5.2 django_app> docker-compose up	42
5.7.1.5.3 docker ps	44

5.7.1.5.4	<code>docker-compose down</code>	44
5.7.1.6	Compose file examples	44
5.7.1.6.1	Compose file examples	44
5.7.2	<code>gitlab ARM</code>	47
5.7.2.1	Overview of Docker Compose	48
6	compose-file	49
6.1	Versions	49
6.1.1	3.7	49
6.1.2	3.6	50
7	Bonnes pratiques Docker	51
7.1	<code>actu.alfa-safety.fr</code>	51
7.2	Best practices for writing Dockerfiles	52
7.2.1	9-pillars-of-containers-best-practices	52
7.3	Best practices for writing Dockerfiles from Nick Janetakis	52
7.3.1	Docker Best practices from Nick Janetakis	52
7.3.1.1	<code>Dockerfile</code>	53
7.3.1.2	<code>docker-compose.yml</code>	53
7.3.1.3	<code>.dockerrcignore</code>	53
7.3.1.4	Example Apps for Popular Web Frameworks	54
7.3.1.5	Flask example	54
7.3.1.5.1	<code>Flask Dockerfile</code>	54
7.3.1.5.2	<code>.env</code> file	54
7.3.1.5.3	<code>Flask docker-compose.yml</code>	55
7.3.1.5.4	<code>hello/app.py</code>	55
7.3.1.5.5	<code>.gitignore</code>	55
8	Docker machine	59
9	Docker swarm	61
9.1	Docker swarm articles	61
9.1.1	Docker swarm articles 2018	61
9.1.1.1	Only one host for production environment. What to use: <code>docker-compose</code> or single node swarm ?	61
9.1.1.1.1	Question	62
9.1.1.1.2	Response	62
10	Docker commands	65
10.1	<code>docker help</code>	66
10.2	<code>docker attach</code>	68
10.3	<code>docker build</code>	68
10.3.1	Description	68
10.3.2	<code>docker build --no-cache</code>	68
10.4	<code>docker commit</code>	68
10.5	<code>docker cp</code>	69
10.6	<code>docker diff</code>	69
10.7	<code>docker exec</code>	69
10.8	<code>docker export</code>	69
10.9	<code>docker history</code>	69
10.10	<code>docker inspect</code>	70
10.11	<code>docker images</code>	70
10.12	<code>docker kill</code>	70
10.13	<code>docker login</code>	70
10.14	<code>docker logs</code>	70

10.14.1 Description	71
10.15 docker ps	71
10.15.1 docker ps –filter	71
10.16 docker pull	71
10.17 docker rename	71
10.18 docker run	71
10.18.1 detach-keys	72
10.18.2 downloading images	72
10.18.3 labels	72
10.18.4 entrypoint	72
10.19 docker search	72
10.19.1 Description	72
10.20 docker system	73
10.20.1 docker system prune	74
10.21 docker stop	74
10.22 docker tag	74
10.23 docker volume	74
11 Dockerfile	75
11.1 Deprecated	75
11.1.1 MAINTAINER	75
12 Docker network	77
12.1 Las networking	77
13 Volumes Docker	79
13.1 Use volumes	79
13.2 Create and manage volumes	80
13.2.1 docker volume create	80
13.2.2 docker volume ls	80
14 Registry	81
14.1 Definition	81
14.2 Understanding image naming	82
14.3 Use cases	82
14.4 Implementations	82
14.4.1 Docker Registry implementations	82
14.4.1.1 Gitlab Container Registry	82
14.4.1.1.1 Historique	83
14.4.1.1.2 Administration	84
14.4.1.1.3 Examples	84
14.4.5 Examples	84
14.4.5.1 Docker Registry examples	84
14.4.5.1.1 Running Your Own Registry	84
15 Glossaire Docker	85
16 docker FAQ	89
16.1 How to delete all your docker images ?	89
16.2 How to run a shell in our running container ?	89
16.3 How to delete stopped containers ?	90
16.4 Where can I find example compose files ?	90
17 Hébergeurs Docker	91
17.1 Gitlab	91

17.2	Amazon	91
18	Docker documentation	93
18.1	Docker aquasec documentation	93
18.1.1	About this Site	93
19	Docker people	95
19.1	Bret Fischer	95
19.1.1	News	95
19.1.1.1	2018	95
19.2	Nick Janetakis	95
19.2.1	Best practices	97
19.3	Mickael Bright	97
19.3.1	Activités septembre 2018 à Grenoble	97
19.4	Stéphane Beuret	97
19.4.1	Activités 2018	99
19.4.1.1	GNU/Linux Mag hors série N°98 septembre 2018	99
19.4.1.2	GNU/linux mag N°217 juillet 2018	99
19.4.1.3	GNU/linux mag N°214 avril 2018	99
19.4.1.4	GNU/linux mag N°211 janvier 2018	99
19.4.1.5	GNU/linux mag N°204 mai 2017	99
20	linux techniques	101
20.1	namespaces	101
20.2	cgroups	101
21	Docker videos	103
21.1	2018	103
22	Actions/news	105
22.1	Actions/news 2018	105
22.1.1	Actions/news 2018-11	105
22.1.1.1	Jeudi 8 novembre 2018 : sortie de Docker 18.09	105
22.1.1.2	Actions/news 2018-09	105
22.1.2.1	Démonstration de Mickael Bright à Grenoble	105
22.1.1.3	Actions/news 2018-08	105
22.1.3.1	~jpetazzo/Dérisquer son infrastructure avec les conteneurs	106
22.1.1.4	Actions/news 2018-06	106
22.1.4.1	Et je suis passé à https avec Docker et Traefik (https://letsencrypt.org)	106
22.1.4.1.1	Conclusion	106
22.1.4.2	Interesting Dockerfile and docker-compose files	107
22.1.4.2.1	Joe Jasinski	107
22.1.4.2.2	Jeff Triplett	107
22.1.4.2.3	cookiecutter-django docker-postgres backups	107
22.1.4.3	Introducing an Easier Way To Design Applications in Docker Desktop	107
22.1.4.4	Docker adoption	107
22.1.5	Actions/news mai 2018	107
22.1.5.1	Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron	108
22.1.5.2	DjangoCon 2018 - An Intro to Docker for Djangonauts by Lacey Williams	108
22.1.5.3	hard-multi-tenancy-in-kubernetes	108
22.1.5.4	containers-security-and-echo-chambers	108
22.1.5.5	Aly Sivji, Joe Jasinski, tathagata dasgupta (t) - Docker for Data Science - PyCon 2018	108
22.1.5.5.1	Description	108
22.1.5.6	Créez un cluster hybride ARM/AMD64 (GNU/Linux N°215 mai 2018)	109

22.1.6 Actions/news avril 2018	109
22.1.6.1 Les slides de Petazzoni pour les formations docker et kubernetes d'avril 2018	109
22.1.6.1.1 Le répertoire source des slides	109
22.1.6.1.2 Autres conférences passées et futures	109
22.1.6.2 Docker for the busy researcher (from Erik Matsen)	109
22.1.6.2.1 Why Docker ?	110
22.1.7 Actions/news mars 2018	110
22.1.7.1 Jeudi 29 mars 2018 : Running Your Own Registry	110
22.1.7.1.1 Docker Registry	110
22.1.7.2 Jeudi 29 mars 2018 : Article de Jérôme Petazzoni : Containers par où commencer ?	110
22.1.8 Actions/news février 2018	110
22.1.8.1 Mardi 13 février 2018: import d'une nouvelle base de données données db_id3_intranet	110
22.1.8.1.1 Suppression du volume djangoid3_intranet_volume (docker volume rm djangoid3_intranet_volume)	111
22.1.8.1.2 Import de la nouvelle base de données (docker-compose -f docker-compose_for_existing_database.yml up --build)	111
22.1.8.1.3 Accès à la nouvelle base de données (docker-compose exec db bash)	113
22.1.8.1.4 Arrêt du service (docker-compose -f .docker-compose_for_existing_database.yml down)	115
22.1.8.2 Mardi 13 février 2018: mise en place d'une base de données PostgreSQL 10.2 avec import de la base de données db_id3_intranet	115
22.1.8.2.1 docker-compose_for_existing_database.yml	115
22.1.8.2.2 Contenu du répertoire init	116
22.1.8.3 Lundi 12 février 2018: mise en place d'une base de données PostgreSQL 10.2	117
22.1.8.3.1 Dockerfile	118
22.1.8.3.2 docker-compose.yml	118
22.1.8.3.3 Accès HeidiSQL à partir de la machine hôte	118
22.1.9 Actions/news janvier 2018	119
22.1.9.1 Mercredi 31 janvier 2018 : export/import d'une base de données PostgreSQL (tutoriel PostgreSQL)	119
22.1.9.1.1 Dockerfile	119
22.1.9.1.2 docker-compose.yml	119
22.1.9.1.3 Export	119
22.1.9.1.4 Import	119
22.1.9.1.5 Commandes docker-compose	119
22.1.9.2 Mercredi 31 janvier 2018 : Bilan mardi 30 janvier 2018	120
22.1.9.2.1 Suppression de la base db_id3_intranet	120
22.1.9.2.2 Bilan mardi 30 janvier 2018	121
22.1.9.2.3 Pour lancer PostgreSQL	122
22.1.9.2.4 Pour accéder au conteneur	122
22.1.9.2.5 Livre <i>PostgreSQL : Administration et exploitation de vos bases de données</i>	123
22.1.9.3 Mardi 30 janvier 2018 : écriture des fichiers Dockerfile et docker-compose.yml	123
22.1.9.3.1 Objectifs pour la journée	123
22.1.9.3.2 Avancement, découverte	123
22.1.9.3.3 Historique	124
22.1.9.4 Lundi 29 janvier 2018 : encore un nouveau tutoriel : A Simple Recipe for Django Development In Docker (Bonus: Testing with Selenium) de Jacob Cook	124
22.1.9.4.1 Analyse et plan de travail pour la journée	124
22.1.9.4.2 Autre projet intéressant	124
22.2 Actions/news 2017	125
22.2.1 Actions/news août 2017	125
22.2.1.1 4 août 2017 “Docker et Shorewall” par Guillaume Cheramy	125
22.2.1.1.1 Créer les règles shorewall pour Docker	126

23 Images Docker (Store Docker, ex Hub docker)	127
23.1 Nouveau: le docker store: https://store.docker.com/	128
23.2 Ancien: le hub docker https://hub.docker.com/explore/	128
23.3 Gitlab registry	128
23.3.1 GitLab Container Registry	128
23.3.1.1 Introduction	128
23.3.1.2 Private registry	130
23.3.1.2.1 Private GitLab Container Registry	130
23.4 Images OS	131
23.4.1 Images Alpine	131
23.4.1.1 Short Description	131
23.4.1.2 Description	131
23.4.1.3 Dockerfile	132
23.4.2 Images Debian	132
23.4.2.1 Short Description	134
23.4.2.2 Description	134
23.4.3 Images Ubuntu	134
23.4.3.1 Short Description	134
23.4.3.2 Description	134
23.4.3.3 La Philosophie d'Ubuntu	136
23.4.4 Images CentOS	136
23.4.4.1 Short Description	138
23.4.4.2 Description	138
23.4.4.3 Structures	138
23.5 Images langages	138
23.5.1 Images Python	138
23.5.1.1 Short Description	140
23.5.1.2 What is Python ?	140
23.5.1.3 How to use this image	140
23.5.2 Images pipenv	140
23.5.2.1 Short Description	141
23.5.2.2 What is Python ?	141
23.5.2.3 Dockerfile	141
23.5.2.4 How to use this image	142
23.5.3 Images PHP	142
23.5.3.1 Short Description	143
23.5.3.2 What is PHP ?	143
23.5.4 Images Ruby	143
23.5.4.1 Short Description	143
23.5.4.2 What is Ruby ?	143
23.5.5 Images Node	144
23.5.5.1 Short Description	144
23.5.5.2 What is Node.js ?	144
23.5.6 Images Go (Golang)	145
23.5.6.1 Short Description	145
23.5.6.2 What is Go ?	145
23.5.7 Images OpenJDK (Java)	146
23.5.7.1 Short Description	146
23.5.7.2 What is OpenJDK ?	146
23.5.7.3 How to use this image	146
23.6 Images webserver : serveurs HTTP (serveurs Web)	148
23.6.1 Images Apache HTTPD	148
23.6.1.1 Short Description	148
23.6.1.2 What is httpd ?	150

23.6.1.3 Configuration	150
23.6.1.4 SSL/HTTPS	150
23.6.2 Images Apache HTTPD bitnami	150
23.6.2.1 Short Description	152
23.6.2.1.1 What is Apache ?	152
23.6.2.2 TL;DR;	152
23.6.2.3 Docker Compose	152
23.6.2.4 Dockerfile	152
23.6.2.5 Why use Bitnami Images ?	153
23.6.2.6 Adding custom virtual hosts	153
23.6.2.6.1 Step 1: Write your my_vhost.conf file with the following content	153
23.6.2.6.2 Step 2: Mount the configuration as a volume	153
23.6.2.7 Using custom SSL certificates	154
23.6.2.7.1 Step 1: Prepare your certificate files	154
23.6.2.7.2 Step 2: Run the Apache image	154
23.6.2.8 Full configuration	155
23.6.2.8.1 Step 1: Run the Apache image	155
23.6.2.8.2 Step 2: Edit the configuration	155
23.6.2.8.3 Step 3: Restart Apache	155
23.6.2.9 Logging	155
23.6.2.10 Upgrade this image	156
23.6.2.10.1 Step 1: Get the updated image	156
23.6.2.10.2 Step 2: Stop and backup the currently running container	156
23.6.2.10.3 Step 3: Remove the currently running container	156
23.6.2.10.4 Step 4: Run the new image	156
23.6.2.11 Notable Changes	157
23.6.2.11.1 2.4.34-r8 (2018-07-24)	157
23.6.2.11.2 2.4.18-r0	157
23.6.2.11.3 2.4.12-4-r01	157
23.6.3 Images apache Tomcat	157
23.6.3.1 Short Description	158
23.6.3.2 What is Apache Tomcat ?	158
23.6.4 Images webserver : serveurs Web + reverse proxy + load balancer	158
23.6.4.1 Apache HTTP Server + mod_proxy	158
23.6.4.2 Nginx	160
23.6.4.2.1 Images nginx (engine-x)	160
23.7 Images db : bases de données	161
23.7.1 Images PostgreSQL	161
23.7.1.1 Short Description	161
23.7.1.2 Description	161
23.7.1.3 What is PostgreSQL ?	163
23.7.1.4 Environment Variables	163
23.7.1.4.1 POSTGRES_PASSWORD	163
23.7.1.4.2 POSTGRES_USER	163
23.7.1.4.3 PGDATA	164
23.7.1.4.4 POSTGRES_DB	164
23.7.1.4.5 POSTGRES_INITDB_WALDIR	164
23.7.1.5 Docker Secrets	164
23.7.1.6 How to extend this image	164
23.7.1.6.1 Extends with a Dockerfile	165
23.7.1.7 docker-compose up	165
23.7.2 Images MariaDB	167
23.7.2.1 Short Description	167
23.7.2.2 What is MariaDB ?	167

23.7.2.3	How to use this image	168
23.7.3	Docker sybase	168
23.8	Images message queue	169
23.8.1	Images rabbitmq	169
23.8.1.1	What is RabbitMQ ?	169
23.8.1.2	Rabbitmq and celery	169
23.9	Images outils collaboratifs	170
23.9.1	Images Gitlab community edition	170
23.9.1.1	Short Description	170
23.9.2	Images Redmine	170
23.9.2.1	Short Description	171
23.9.3	Images Wordpress	171
23.9.3.1	Short Description	171
23.10	Images “documentation”	172
23.10.1	Images MiKTeX	172
23.10.1.1	Short Description	172
23.11	Images outils scientifiques	173
23.11.1	Images Anaconda3	173
23.11.1.1	Short Description	173
23.11.1.2	Usage	173
23.12	Images apprentissage	175
23.12.1	Image dockersamples/static-site	175
23.12.2	Image hello world	175
23.12.2.1	Short Description	175
24	Tutoriels Docker	177
24.1	Avril 2018 container training from Jérôme Petazzoni	177
24.1.1	Intro Avril 2018	177
24.1.1.1	Overview	178
24.1.1.1.1	A brief introduction	178
24.1.1.1.2	About these slides	179
24.1.1.1.3	Docker 30,000ft overview	179
24.1.1.1.4	OK... Why the buzz around containers ?	179
24.1.1.1.5	Deployment becomes very complex	180
24.1.1.1.6	Results	180
24.1.1.1.7	Escape dependency hell	180
24.1.1.1.8	On-board developers and contributors rapidly	181
24.1.1.1.9	Implement reliable CI easily	181
24.1.1.1.10	Use container images as build artefacts	181
24.1.1.1.11	Decouple “plumbing” from application logic	182
24.1.1.1.12	Formats and APIs, before Docker	182
24.1.1.1.13	Formats and APIs, after Docker	182
24.1.1.1.14	Shipping, before Docker	182
24.1.1.1.15	Shipping, after Docker	183
24.1.1.1.16	Example	183
24.1.1.1.17	Devs vs Ops, before Docker	183
24.1.1.1.18	Devs vs Ops, after Docker	183
24.1.1.2	History of containers ... and Docker	184
24.1.1.2.1	First experimentations	184
24.1.1.2.2	The VPS age (until 2007-2008)	185
24.1.1.2.3	Containers = cheaper than VMs	185
24.1.1.2.4	The PAAS period (2008-2013)	185
24.1.1.2.5	Containers = easier than VMs	185
24.1.1.2.6	First public release of Docker	186

24.1.1.2.7	Docker early days (2013-2014)	186
24.1.1.2.8	First users of Docker	187
24.1.1.2.9	Positive feedback loop	187
24.1.1.2.10	Maturity (2015-2016)	187
24.1.1.2.11	Docker becomes an industry standard	187
24.1.1.2.12	Docker becomes a platform	187
24.1.2	Chapter1 Avril 2018	188
24.1.3	Chapter2 Avril 2018 container training	188
24.1.3.1	Our first containers	188
24.1.3.1.1	Hello World	189
24.1.3.1.2	Starting another container	189
24.1.3.2	Background containers	189
24.1.3.2.1	Objectives	189
24.1.3.2.2	A non-interactive container	190
24.1.3.2.3	Run a container in the background	190
24.1.3.2.4	List running containers	190
24.1.3.2.5	View only the IDs of the containers	191
24.1.3.2.6	Combining flags	191
24.1.3.2.7	View the logs of a container	192
24.1.3.2.8	View only the tail of the logs	192
24.1.3.2.9	Follow the logs in real time	192
24.1.3.2.10	Stop our container	193
24.1.3.2.11	Stopping our containers	193
24.1.3.2.12	Killing the remaining containers	194
24.1.3.3	Restarting and attaching to containers	194
24.1.3.3.1	Introduction	194
24.1.3.3.2	Background and foreground	195
24.1.3.3.3	Detaching from a container	195
24.1.3.3.4	Specifying a custom detach sequence	195
24.1.3.3.5	Attaching to a container	196
24.1.3.3.6	Detaching from non-interactive containers	196
24.1.3.3.7	Restarting a container	196
24.1.3.4	Understanding Docker images	197
24.1.3.4.1	Objectives	197
24.1.3.4.2	What is an image ?	198
24.1.3.4.3	Differences between containers and images	198
24.1.3.4.4	Object-oriented programming	198
24.1.3.4.5	Wait a minute	199
24.1.3.4.6	Creating the first images	199
24.1.3.4.7	Creating other images	199
24.1.3.4.8	Images namespaces	200
24.1.3.4.9	Root namespace	200
24.1.3.4.10	User namespace	200
24.1.3.4.11	Self-Hosted namespace	201
24.1.3.4.12	How do you store and manage images ?	201
24.1.3.4.13	Showing current images	201
24.1.3.4.14	Searching for images	202
24.1.3.4.15	Downloading images	203
24.1.3.4.16	Pulling an image	203
24.1.3.4.17	Image and tags	203
24.1.3.4.18	When to (not) use tags	204
24.1.3.4.19	Section summary	204
24.1.4	Chapter3 Avril 2018	204
24.1.4.1	Building images interactively	204

24.1.4.1.1	Building images interactively	205
24.1.4.1.2	The plan	205
24.1.4.1.3	Setting up our container	206
24.1.4.1.4	Inspect the changes	207
24.1.4.1.5	Docker tracks filesystem changes	207
24.1.4.1.6	Copy-on-write security benefits	208
24.1.4.1.7	Commit our changes into a new image	208
24.1.4.1.8	Testing our new image	208
24.1.4.1.9	Tagging images	209
24.1.4.1.10	What's next ?	209
24.1.4.2	Building Docker images with a Dockerfile	209
24.1.4.2.1	Objectives	210
24.1.4.2.2	Dockerfile overview	210
24.1.4.2.3	Writing our first Dockerfile	210
24.1.4.2.4	Type this into our Dockerfile	211
24.1.4.2.5	Build it!	211
24.1.4.2.6	What happens when we build the image ?	211
24.1.4.2.7	Sending the build context to Docker	212
24.1.4.2.8	Executing each step	213
24.1.4.2.9	The caching system	213
24.1.4.2.10	Running the image	213
24.1.4.2.11	Using image and viewing history	213
24.1.4.2.12	Introducing JSON syntax	214
24.1.4.2.13	JSON syntax vs string syntax	214
24.1.4.2.14	When to use JSON syntax and string syntax	214
24.1.4.3	CMD and ENTRYPOINT	215
24.1.4.3.1	Objectives	216
24.1.4.3.2	Defining a default command	216
24.1.4.3.3	Adding CMD to our Dockerfile	216
24.1.4.3.4	Build and test our image	216
24.1.4.3.5	Overriding CMD	217
24.1.4.3.6	Using ENTRYPOINT	217
24.1.4.3.7	Adding ENTRYPOINT to our Dockerfile	217
24.1.4.3.8	Implications of JSON vs string syntax	218
24.1.4.3.9	Build and test our image	218
24.1.4.3.10	Using CMD and ENTRYPOINT together	218
24.1.4.3.11	CMD and ENTRYPOINT together	219
24.1.4.3.12	Build and test our image	219
24.1.4.3.13	Overriding the image default parameters	220
24.1.4.3.14	Overriding ENTRYPOINT	220
24.1.4.4	Copying files during the build	220
24.1.4.4.1	Objectives	221
24.1.4.4.2	Build some C code	221
24.1.4.4.3	The Dockerfile	221
24.1.4.4.4	Testing our C program	221
24.1.4.4.5	COPY and the build cache	222
24.1.4.4.6	Details	222
24.1.4.4.7	Next step : multi-stage building	223
24.1.4.5	Multi-stage builds	223
24.1.4.5.1	Multi-stage builds	223
24.1.4.5.2	Multi-stage builds principles	224
24.1.4.5.3	Multi-stage builds in practice	224
24.1.4.5.4	Multi-stage builds for our C program	224
24.1.4.5.5	Multi-stage build Dockerfile	224

24.1.4.5.6 Comparing single/multi-stage build image sizes	226
24.1.4.6 Publishing images to the Docker Hub	226
24.1.4.6.1 Publishing images to the Docker Hub	226
24.1.4.6.2 Logging into our Docker Hub account	227
24.1.4.6.3 Image tags and registry addresses	227
24.1.4.6.4 Image tags and registry addresses	227
24.1.4.6.5 Tagging an image to push it on the Hub	228
24.1.4.7 Tips for efficient Dockerfiles	228
24.1.4.7.1 Tips for efficient Dockerfiles	229
24.1.4.7.2 Reducing the number of layers	229
24.1.4.7.3 Avoid re-installing dependencies at each build	230
24.1.4.7.4 Example “bad” Dockerfile	230
24.1.4.7.5 Fixed Dockerfile	230
24.1.4.7.6 Embedding unit tests in the build process	231
24.1.5 Chapter4 Avril 2018	231
24.1.5.1 Naming and inspecting containers	231
24.1.5.1.1 Objectives	232
24.1.5.1.2 Naming our containers	232
24.1.5.1.3 Default names	232
24.1.5.1.4 Specifying a name	233
24.1.5.1.5 Renaming containers	233
24.1.5.1.6 Inspecting a container	233
24.1.5.1.7 Parsing JSON with the Shell	233
24.1.5.1.8 Using –format	234
24.1.5.2 Naming and inspecting containers	234
24.1.5.2.1 Labels	234
24.1.5.2.2 Using labels	235
24.1.5.2.3 Querying labels	235
24.1.5.2.4 Using labels to select containers (docker ps –filter)	235
24.1.5.2.5 Use-cases for labels	236
24.1.5.3 Getting inside a container	236
24.1.5.3.1 Objectives	237
24.1.5.3.2 Getting a shell	237
24.1.5.3.3 Not getting a shell	237
24.1.5.3.4 Viewing container processes from the host	237
24.1.5.3.5 What’s the difference between a container process and a host process ?	238
24.1.5.3.6 Getting a shell in a running container	238
24.1.5.3.7 Caveats	238
24.1.5.3.8 Getting a shell in a stopped container	239
24.1.5.3.9 Analyzing a stopped container	239
24.1.5.3.10 Viewing filesystem changes	239
24.1.5.3.11 Accessing files	240
24.1.5.3.12 Exploring a crashed container (docker commit + docker run –ti –entrypoint)	240
24.1.5.3.13 Obtaining a complete dump (docker export)	240
24.1.5.4 Container networking basics	241
24.1.5.5 Container network drivers	241
24.1.5.6 Container network model	241
24.1.5.7 Service discovery with containers	241
24.1.5.8 Ambassadors	242
24.1.6 Chapter5 Avril 2018	242
24.1.6.1 Local development workflow with Docker	242
24.1.6.1.1 Objectives	243
24.1.6.1.2 Containerized local development environments	243
24.1.6.1.3 Working on the “namer” application	243

24.1.6.1.4	Looking at the code	243
24.1.6.2	Working with volumes	244
24.1.6.2.1	Objectives	244
24.1.6.2.2	Working with volumes	245
24.1.6.2.3	Volumes are special directories in a container	245
24.1.6.2.4	Volumes bypass the copy-on-write system	245
24.1.6.2.5	Volumes can be shared across containers	246
24.1.6.2.6	Sharing app server logs with another container	246
24.1.6.2.7	Volumes exist independently of containers	246
24.1.6.2.8	Naming volumes	247
24.1.6.2.9	Using our named volumes	247
24.1.6.2.10	Using a volume in another container	248
24.1.6.2.11	Managing volumes explicitly	248
24.1.6.2.12	Migrating data with –volumes-from	249
24.1.6.2.13	Data migration in practice	249
24.1.6.2.14	Upgrading Redis	249
24.1.6.2.15	Testing the new Redis	250
24.1.6.2.16	Volumes lifecycle	250
24.1.6.2.17	Checking volumes defined by an image	250
24.1.6.2.18	Checking volumes used by a container	250
24.1.6.2.19	Sharing a single file	251
24.1.6.2.20	Volume plugins	251
24.1.6.2.21	Volumes vs. Mounts	252
24.1.6.2.22	–mount syntax	252
24.1.6.2.23	Section summary	252
24.1.6.3	Compose for development stacks	252
24.1.6.3.1	Compose for development stacks	253
24.1.6.3.2	What is Docker Compose ?	253
24.1.6.3.3	Compose overview	254
24.1.6.3.4	Checking if Compose is installed	254
24.1.6.3.5	Launching Our First Stack with Compose	254
24.1.6.3.6	Launching Our First Stack with Compose	258
24.1.6.3.7	Stopping the app	258
24.1.6.3.8	The docker-compose.yml file	259
24.1.6.3.9	Compose file versions	259
24.1.6.3.10	Containers in docker-compose.yml	260
24.1.6.3.11	Container parameters	260
24.1.6.3.12	Compose commands	260
24.1.6.3.13	Check container status	261
24.1.6.3.14	Cleaning up (1)	261
24.1.6.3.15	Cleaning up (2)	261
24.1.6.3.16	Special handling of volumes	262
24.1.6.3.17	Compose project name	262
24.1.6.3.18	Running two copies of the same app	262
24.1.6.4	Managing hosts with Docker Machine	263
24.1.7	Chapter6 Avril 2018	263
24.1.8	Chapter7 Avril 2018	263
24.1.9	Chapter8 Avril 2018	263
24.2	Les conseils et formations de Jérôme Petazzoni	263
24.2.1	Se former, seul ou accompagné	264
24.2.2	Jérôme Petazzoni Container training	264
24.2.3	Jérémy Garrouste	265
24.2.4	Les slides de la formation d'avril 2018	265
24.3	Tutoriels Docker pour Windows	265

24.3.1	Installation	265
24.3.2	docker –version	265
24.3.3	docker-compose –version	267
24.3.4	docker-machine –version	267
24.3.5	notary version	267
24.3.6	Binaires docker sous Windows 10	267
24.3.7	Where to go next	267
24.4	Get started (https://docs.docker.com/get-started/)	268
24.4.1	docker run hello-world	268
24.4.2	docker –version	269
24.4.3	Conclusion	269
24.4.4	Parts	269
24.4.4.1	Get started Part2 : Containers	269
24.4.4.1.1	Prérequis	269
24.4.4.1.2	Build the app: docker build -t friendlyhello	270
24.4.4.1.3	docker images	271
24.4.4.1.4	Run the app: docker run -p 4000:80 friendlyhello	272
24.4.4.1.5	docker container ls	272
24.4.4.1.6	docker container stop 06193b763075	272
24.4.4.1.7	Tag the image: docker tag friendlyhello id3pvergain/get-started:part2	272
24.4.4.1.8	Publish the image	272
24.4.4.1.9	Pull and run the image from the remote repository	273
24.4.4.2	Get started Part3 : services	274
24.4.4.2.1	Prerequisites	274
24.4.4.2.2	Introduction	275
24.4.4.2.3	About services	275
24.4.4.2.4	Your first docker-compose.yml file	275
24.4.4.2.5	Run your new load-balanced app	276
24.4.4.2.6	docker swarm init	276
24.4.4.2.7	docker stack deploy -c docker-compose.yml getstartedlab	276
24.4.4.2.8	docker service ls	277
24.4.4.2.9	docker service ps getstartedlab_web	277
24.4.4.2.10	docker container ls -q	277
24.4.4.2.11	Sous WSL (Windows Subsystem Linux)	278
24.4.4.2.12	Scale the app	278
24.4.4.2.13	Take down the app (docker stack rm getstartedlab)	278
24.4.4.2.14	Take down the swarm (docker swarm leave –force)	278
24.4.4.3	Get started Part4 : swarms	279
24.4.4.3.1	Introduction	279
24.4.4.3.2	Understanding Swarm clusters	280
24.4.4.3.3	Set up your swarm	280
24.4.4.3.4	Encore Bloqué	280
24.5	A Simple Recipe for Django Development In Docker par Adam King (Advanced tutorial)	281
24.5.1	Dockerfile Adam King	282
24.5.1.1	WORKDIR	283
24.5.2	docker-compose.yml Adam King	283
24.5.2.1	stdin_open: true, tty:true	283
24.5.2.2	docker-compose up -d	283
24.5.3	Explore your container (docker-compose exec django bash)	284
24.5.4	Take a break	284
24.5.5	Next Steps: Add a MySQL Database	284
24.5.5.1	db	285
24.5.5.1.1	MYSQL_ROOT_PASSWORD	285
24.5.5.2	DATABASE_URL	285

24.6	Modern DevOps with Django par Jacob Cook (Advanced tutorial)	286
24.6.1	tree	286
24.6.2	Dockerfile Jacob Cook	287
24.6.3	docker-compose.yml Jacob Cook	288
24.6.4	Testing and Production	288
24.6.4.1	docker-compose.test.yml	289
24.6.4.2	docker-compose.staging.yml	289
24.6.4.3	docker-compose.prod.yml	290
24.7	Django for beginners par William Vincent	291
24.7.1	Thanks to William Vincent !	292
24.7.2	tree ch4-message-board-app	293
24.7.3	Installing django with pipenv and python 3.6	293
24.7.3.1	Dockerfile	294
24.7.3.2	Pipfile	294
24.7.4	docker build –tag gdevops/django36_ch4	294
24.7.5	docker images	296
24.7.6	mb_project/settings.py	296
24.7.7	Launch the db and web services with docker-compose.yml	296
24.7.7.1	db	297
24.7.7.2	web	297
24.7.7.3	volumes	297
24.7.7.4	ports	297
24.7.7.5	volumes	297
24.7.8	docker-compose run web python /code/manage.py migrate –noinput	297
24.7.9	docker-compose run web python /code/manage.py createsuperuser	299
24.7.10	docker-compose up	299
24.7.11	docker-compose ps	302
24.7.12	docker-compose exec db bash	302
24.7.13	psql -d db -U postgres	302
24.7.13.1	dt	303
24.7.13.2	conninfo	303
24.7.13.3	dn	303
24.7.13.4	d posts_post	304
24.8	A Brief Intro to Docker for Djangonauts par Lacey Williams	304
24.8.1	Introduction	305
24.8.2	Annonce de l’écriture du tutoriel le 20 octobre 2017	305
24.8.3	Dockerfile Lacey Williams	305
24.8.3.1	FROM python:3.6	305
24.8.3.2	ENV PYTHONUNBUFFERED 1	305
24.8.3.3	ENV DJANGO_ENV dev	305
24.8.3.4	ENV DOCKER_CONTAINER 1	309
24.8.3.5	EXPOSE 8000	309
24.8.4	docker-compose.yml Lacey Williams	309
24.8.5	version: ‘3’	310
24.8.6	services	310
24.8.6.1	db	310
24.8.6.1.1	volumes	310
24.8.6.2	web	310
24.8.6.2.1	build	310
24.8.6.2.2	command: python /code/manage.py migrate –noinput	311
24.8.6.2.3	command: python /code/manage.py runserver 0.0.0.0:8000	311
24.9	Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron	311
24.9.1	Format PDF	311
24.9.2	Introduction	311

24.9.3 Conclusion	312
24.10 Docker: les bons réflexes à adopter par Paul MARS (MISC 95)	312
24.10.1 Dockerfile MISC 95	313
24.10.2 Fichiers .env	313
24.11 Tutoriel Django step by step	313
24.12 Tutoriel erroneousboat Docker Django	314
24.12.1 tree	314
24.12.2 docker-compose.yml	315
24.13 Tutoriel Utilisation de pipenv avec Docker	316
24.13.1 Les fichiers	316
24.13.2 Réécriture du fichier Dockerfile	317
24.13.3 app.py	317
24.13.4 docker build -t docker-pipenv-sample : construction de l'image	317
24.13.5 docker run -p 5000:5000 docker-pipenv-sample	319
24.13.6 http://localhost:5000/	319
24.13.7 docker ps	319
24.13.8 docker exec -it 1a0a3dc7924d bash	320
24.13.9 docker rm 1a0a3dc7924d: suppression du conteneur à l'arrêt	320
24.13.10 docker rmi docker-pipenv-sample: suppression de l'image	320
24.14 play with docker	320
24.14.1 Docker for IT Pros and System Administrators Stage 1	321
24.14.2 Docker for Beginners - Linux	321
24.15 Centos7	321
24.15.1 Plan de travail	322
24.15.2 yum update	322
24.15.3 yum install -y https://centos7.iuscommunity.org/ius-release.rpm	326
24.15.4 yum install -y python36u python36u-libs python36u-devel python36u-pip	328
24.15.5 python3.6	330
24.15.6 yum install which	330
24.15.7 which pip3.6	331
24.15.8 docker build -t id3centos7:1	332
24.15.9 docker images	339
24.15.10 docker run --name test -it id3centos7:1	340
24.15.11 Problème avec regex	340
24.15.12 yum install gcc	341
24.15.13 yum install openldap-devel	344
24.15.14 pip install pyldap	348
24.15.15 Nouveau fichier Dockerfile	348
24.15.15.1 Dockerfile	348
24.15.15.2 which python3.6	348
24.15.15.3 python3.6 -m pip install pipenv	348
24.15.16 Nouveau Dockerfile	349
24.15.16.1 Dockerfile	349
24.15.16.2 docker build -t id3centos7:0.1.1	350
24.15.17 Nouveau fichier Dockerfile	360
24.15.17.1 Dockerfile	360
24.15.17.2 Construction de l'image docker build -t id3centos7:0.1.2	360
24.15.17.3 docker run --name id3centos7.1.2 -it id3centos7:0.1.2	361
24.15.18 Nouveau dockerfile	361
24.15.18.1 Dockerfile	361
24.15.19 Nouveau fichier Dockerfile	363
24.15.19.1 Dockerfile	363
24.15.20 Nouveau fichier Dockerfile	364
24.16 Tutoriel Docker et Postgresql	368

24.16.1 Modèle de fichier docker-compose.yml	369
24.16.2 docker-compose up	369
24.16.3 docker-compose run postgres psql -h postgres -U postgres	372
24.16.4 docker-compose down	373
24.16.5 docker-compose build	373
24.16.6 docker-compose up	373
24.16.7 docker-compose exec -u postgres db psql	374
24.16.8 docker ps	376
24.16.9 docker exec -it d205b9239366 bash	376
24.16.10 Mardi 30 janvier 2018	376
24.16.10.1 docker-compose.yml	377
24.16.10.2 docker volume ls	378
24.16.10.3 docker volume inspect postgresql_volume_intranet	378
24.16.10.4 docker exec -it 47501acda106 bash	378
24.16.10.5 psql -U postgres	378
24.16.10.6 (liste des bases de données)	379
24.16.10.7 CREATE USER id3admin WITH PASSWORD 'id338';	379
24.16.10.8 CREATE DATABASE db_id3_intranet WITH OWNER = id3admin ENCODING = 'UTF8' CONNECTION LIMIT = -1;	379
24.16.10.9	379
24.16.10.10 docker-compose run db env	380
24.16.10.11 docker-compose config	380
24.16.11 Import de la base de données	380
24.16.12 Mercredi 31 janvier 2018 : export/import d'une base de données PostgreSQL (tutoriel PostgreSQL)	381
24.16.12.1 pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet	381
24.16.12.2 Entête de db.dump	381
24.16.12.3 Expérience substitution de db_id3_save à db_id3_intranet	382
24.16.12.4 psql -U postgres -f db.dump.sql	383
24.16.12.5 docker-compose stop	384
24.16.12.6 docker-compose build	384
24.16.13 CREATE DATABASE db_id3_save WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE = 'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';	385
24.17 Docker OpenLDAP	385
25 Exemples Docker labs	387
25.1 Samples Docker labs	387
25.1.1 Samples Docker labs beginner	387
25.1.1.1 Setup	388
25.1.1.2 docker run hello-world	388
25.1.1.2.1 hello.c	388
25.1.1.2.2 Dockerfile.build	390
25.1.1.3 Running your first container : docker pull alpine	391
25.1.1.3.1 docker pull alpine	391
25.1.1.3.2 docker images	392
25.1.1.3.3 docker run alpine ls -l	392
25.1.1.3.4 docker ps -a	393
25.1.1.3.5 docker run -it alpine /bin/sh	393
25.1.1.4 docker run --help	394
25.1.1.5 docker inspect alpine	397
25.1.1.6 Next Steps: 2.0 Webapps with Docker	399
25.1.2 2) Webapps with Docker (Python + Flask)	399
25.1.2.1 Introduction	401
25.1.2.2 Run a static website in a container : docker run -d dockersamples/static-site	401

25.1.2.3 docker images	401
25.1.2.4 docker run –name static-site -e AUTHOR=”patrick.vergain” -d -P dockersamples/static-site	402
25.1.2.5 docker port static-site	402
25.1.2.6 docker run –name static-site-2 -e AUTHOR=”patrick.vergain” -d -p 8888:80 dockersamples/static-site	402
25.1.2.7 docker stop static-site	404
25.1.2.8 docker rm static-site	405
25.1.2.9 Let’s use a shortcut to remove the second site: docker rm -f static-site-2	405
25.1.2.10 Docker Images	405
25.1.2.11 docker pull ubuntu:16.04	406
25.1.2.12 Create your first image	407
25.1.2.13 Create a Python Flask app that displays random cat pix	407
25.1.2.13.1 app.py	407
25.1.2.13.2 requirements.txt	408
25.1.2.13.3 templates/index.html	408
25.1.2.13.4 Write a Dockerfile	409
25.1.2.13.5 Build the image (docker build -t id3pvergain/myfirstapp)	411
25.1.2.13.6 docker images	412
25.1.2.13.7 Run your image (docker run -p 8888:5000 –name myfirstapp id3pvergain/myfirstapp)	413
25.1.2.13.8 Push your image (docker push id3pvergain/myfirstapp)	413
25.1.2.13.9 docker rm -f myfirstapp	415
25.1.2.13.10 docker ps	415
25.1.2.14 Dockerfile commands summary	416
25.1.2.14.1 FROM	416
25.1.2.14.2 RUN	416
25.1.2.14.3 COPY	416
25.1.2.14.4 CMD	416
25.1.2.14.5 EXPOSE	416
25.1.2.14.6 PUSH	417
25.1.2.15 Next Steps : Deploying an app to a Swarm	417
25.1.3 3.0) Deploying an app to a Swarm	417
25.1.3.1 Introduction	418
25.1.3.2 Voting app	418
25.1.3.3 Deploying the app	418
25.1.3.3.1 docker swarm init	419
25.1.3.3.2 Docker compose file : docker-stack.yml	419
25.1.3.3.3 docker stack deploy –compose-file docker-stack.yml vote	421
25.1.3.3.4 docker stack services vote	421
25.1.3.3.5 Analyse du fichier Docker compose file : docker-stack.yml	422
25.1.3.4 Customize the app	425
25.1.3.4.1 Change the images used	425
25.1.3.4.2 Redeploy: docker stack deploy –compose-file docker-stack.yml vote	427
25.1.3.4.3 Another test run	427
25.1.3.4.4 Remove the stack	427
25.1.3.5 Next steps	429
25.2 Exemples sur Windows 10	429



Fig. 1: Logo Docker <https://www.docker.com>

[https://fr.wikipedia.org/wiki/Docker_\(logiciel\)](https://fr.wikipedia.org/wiki/Docker_(logiciel))

See also:

- https://gitlab.com/gdevops/tuto_docker
- https://gdevops.gitlab.io/tuto_docker/
- <https://docs.docker.com/>
- <https://twitter.com/Docker/lists/docker-captains/members>
- <https://twitter.com/docker>
- <https://plus.google.com/communities/108146856671494713993>
- <http://training.play-with-docker.com/>
- <https://avril2018.container.training/>
- <https://github.com/jpetazzo/container.training>
- <https://hub.docker.com/u/id3pvergain/>

See also:

- <https://jpetazzo.github.io/2018/03/28/containers-par-ou-commencer/>

CHAPTER 1

Introduction à Docker



Fig. 1: Le logo Docker

See also:

- [https://fr.wikipedia.org/wiki/Docker_\(logiciel\)](https://fr.wikipedia.org/wiki/Docker_(logiciel))
- <https://www.docker.com/>
- <https://twitter.com/docker>
- https://en.wikipedia.org/wiki/Cloud_computing
- <https://fr.wikipedia.org/wiki/Virtualisation>
- <https://en.wikipedia.org/wiki/Devops>
- <https://docs.docker.com/get-started/>

Contents

- *Introduction à Docker*
 - *Pourquoi utiliser docker ?*
 - * *Transformation de la DSI des entreprises*

- * Pour donner davantage d'autonomie aux développeurs
- * Faire évoluer son système d'information
- * Pour que ça fonctionne aussi sur une autre machine
- * Livre blanc Ubuntu
- Définitions concernant l'agilité et le mouvement **Devops**
 - * Définition de **Devops** p.34 *Programmez!* p.214 janvier 2018
 - * Définition 2, *Le Devops pour répondre à l'appel de l'innovation* 2018-01-04
 - * Définition 3, extrait p.53 *MISC N95, Janvier/février, 2018, “Ne pas prévoir, c'est déjà gémir”*
 - Citations
 - *Ne pas prévoir, c'est déjà gémir*
 - *La vie, c'est comme une bicyclette, il faut avancer pour ne pas perdre l'équilibre*
 - * *Devops, intégration et déploiement continu, pourquoi est-ce capital et comment y aller ?*
 - * *Agilité et Devops: Extrait p. 35 de [Programmez!], N°214, janvier 2018*
 - * *What is a DevOps Engineer ?*
- Définitions concernant Docker
 - * Définition de Docker sur Wikipedia en français
 - * Docker est “agile”
 - * Docker est portable
 - * Docker est sécurisé
 - * Les conteneurs Docker sont plus légers et rapides que les machines virtuelles
 - Containers
 - Virtual machines (VMs)
 - Docker can run your applications in production at native speed
- Dossier Docker dans le dossier MISC N°95 de janvier/février 2018

1.1 Pourquoi utiliser docker ?

1.1.1 Transformation de la DSI des entreprises

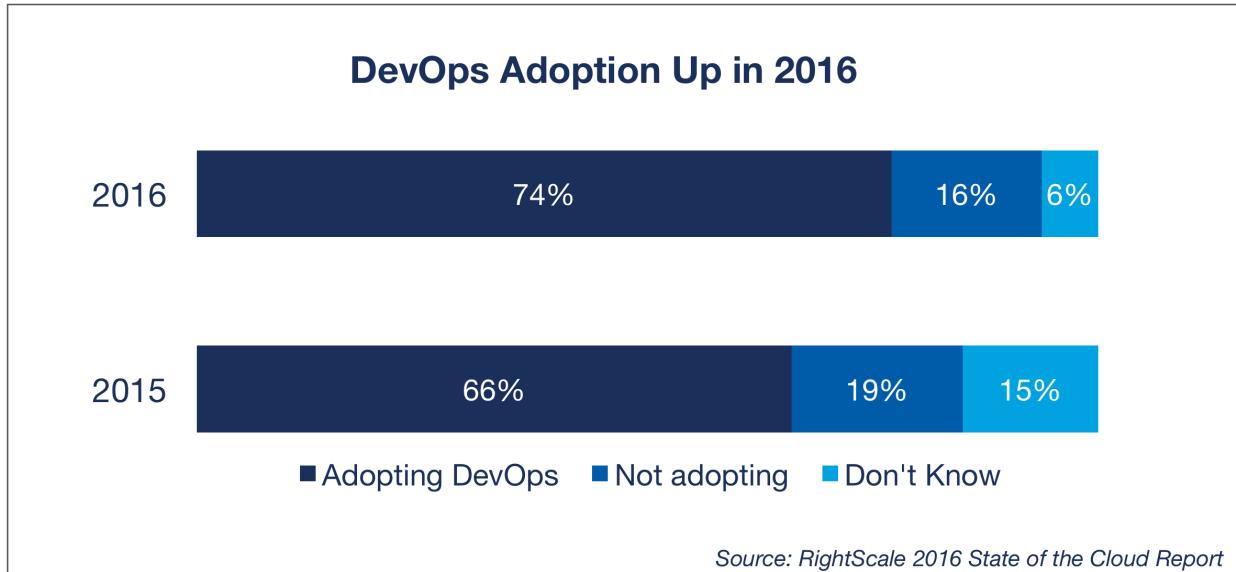
See also:

- <https://actu.alfa-safety.fr/service-aws/devops-et-transformation-dela-dsi/>

Trois évolutions majeures convergent depuis peu et poussent à une transformation de la DSI des entreprises:

- la pression du time to market : l'accélération du rythme d'évolution des applications, en particulier web, pour sortir au plus vite de nouvelles fonctionnalités et répondre aux besoins du marché
- Le Devops : pour livrer plus vite, les équipes de Dev font évoluer leurs méthodes pour rendre les déploiements plus rapides, fréquents et fluides, et attendent que l'infrastructure, les « Ops » évoluent au même rythme.

- le cloud public : il a atteint un niveau de maturité et d'efficacité tel qu'une majorité des DSI travaille maintenant à l'intégrer, souvent sous la pression des équipes de développement,



1.1.2 Pour donner davantage d'autonomie aux développeurs

See also:

- <https://actu.alfa-safety.fr/devops/docker-en-production/>

Avec Docker, donnez davantage d'autonomie aux développeurs

L'un des atouts du conteneur est de donner davantage d'autonomie au développeur. Ce dernier doit pouvoir travailler sur son application sans se soucier de la configuration de la machine sur laquelle il travaille : il doit pouvoir développer sur son poste de travail et pousser son conteneur sur un serveur de test, puis pré-production, et jusqu'en production sans rencontrer de difficultés.

Le développeur doit aussi pouvoir modifier son docker et en gérer les versions sans se préoccuper des conséquences pour la production.

En résumé, un des bénéfices du conteneur c'est qu'il doit pouvoir se déployer n'importe où en toute sécurité.

1.1.3 Faire évoluer son système d'information

See also:

- <https://linuxfr.org/forums/linux-general/posts/docker-en-prod>

Bonjour à tous, après la virtualisation il y a docker (qui a le vent en poupe). Je me dis qu'il y a peut-être quelque chose à faire. Le concept est assez simple, l'utilisation a l'air souple.

Comme par hasard je dois migrer le serveur intranet de ma boite, actuellement il est en RHE 5.x et depuis la version 6.5 docker est intégré par RedHat. Il sert à plusieurs choses :

- dev pour les sites internet;
- PIM interne
- Cacti

- ...

J'aimerais bien avoir un environnement qui me permette d'ajouter Ruby par exemple sans tout péter sur les autres devs, ou installer la version de php 7 alors que le reste doit rester en php 5, la lib rrdtool 1.4 alors qu'un autre doit rester en 1.2... Enfin le genre de chose **bien prise de tête à gérer**.

Après avoir lu pas mal de doc autres que celles de RH je me rend compte qu'à chaque fois se sont des environnements de dev qui sont mis en place mais jamais de la prod, du vrai, du concret, avec des users bien bourrin.

Avez-vous des exemples ou des expériences (réussi ou pas) d'archi en prod ?

1.1.4 Pour que ça fonctionne aussi sur une autre machine

See also:

- <http://putaindecode.io/fr/articles/docker/>

Il était une fois un jeune développeur qui codait tranquillement sur son ordinateur. Il était pressé car comme tout étudiant qui se respecte il devait présenter son travail le lendemain matin. Après des heures de travail, l'application était là, et elle fonctionnait à merveille ! Le lendemain, notre codeur arriva tout fier pour sa présentation, avec son projet sur une clé usb. Il le transfère sur l'ordinateur de son pote et là, ça ne fonctionne pas !

Quel est le problème ?

L'application de notre jeune développeur ne fonctionne pas sur l'ordinateur de son ami à cause d'un problème d'environnement. Entre deux systèmes, il peut y avoir des différences de version sur les dépendances ou encore des bibliothèques manquantes.

1.1.5 Livre blanc Ubuntu

[ubuntu/WP_The_no-nonsense-way-to-accelerate-your-business-with_containers.pdf](#)

1.2 Définitions concernant l'agilité et le mouvement Devops

1.2.1 Définition de Devops p.34 Programmez! p.214 janvier 2018

See also:

- <https://en.wikipedia.org/wiki/Devops>
- <http://david.monniaux.free.fr/dotclear/index.php/post/2018/01/05/Pourquoi-l-informatique-devient-incompr%C3%A9hensible-%C3%A9t-l-impact-sur-la-s%C3%A9curit%C3%A9>

Si le mouvement **Devops** fait bien référence à l'automatisation des tests unitaires ou fonctionnels avec la mise en place de l'intégration continue ou à l'automatisation, ce n'est pas l'aspect principal qu'évoque le mouvement **Devops**.

Le **Devops** est un mouvement qui privilégie la mise en place d'un alignement de l'ensemble de la DSI autour **d'objectifs communs**; le terme **Devops** est la concaténation de dev pour développeur et ops pour opérationnels, soit les ingénieurs responsables des infrastructures.

Avoir une équipe enfermée dans une pièce totalement isolée des équipes de développement pour mettre en place des solutions d'intégration continue ou de livraison continue ne correspond pas à ce concept **Devops**. C'est pourtant cette façon de faire que nous voyons de plus en plus aujourd'hui.

1.2.2 Définition 2, Le Devops pour répondre à l'appel de l'innovation 2018-01-04

See also:

- <https://www.programmez.com/avis-experts/le-Devops-pour-repondre-lappel-de-linnovation-26954>

Le **Devops** est axé sur la collaboration, nécessaire pour développer, tester et déployer des applications rapidement et régulièrement.

C'est un changement culturel, qui met l'accent sur le renforcement de la communication et de la collaboration entre différentes équipes, telles que celles chargées du développement, de l'exploitation et de l'assurance-qualité.

L'objectif est de décloisonner les services qui composent une organisation afin de créer un lieu de travail plus collaboratif et de créer ainsi une synergie qui, en bout de chaîne, profite à l'utilisateur final. Car c'est un fait avéré, la création et la conservation de relations solides avec les clients offrent des avantages exponentiels, dont une diminution de la perte de clientèle et des sources de revenus potentiellement plus nombreuses.

Car le **Devops** est avant tout un concept, il n'existe pas UN outil de **Devops** à proprement parler, mais un faisceau d'outils permettant d'instaurer et d'entretenir une culture **Devops**. Il regroupe à la fois des outils open source et propriétaires dédiés à des tâches spécifiques dans les processus de développement et de déploiement.

D'ailleurs, en parlant de processus, évoquons un instant le déploiement continu.

Le déploiement continu repose entièrement sur des processus, et l'automatisation y joue un rôle clé. Les processus de déploiement continu sont l'un des éléments fondamentaux d'une transformation **Devops**. Le déploiement continu et le **Devops** permettent aux équipes de développement d'accélérer considérablement la livraison de logiciels. Grâce aux processus de déploiement continu et à la culture **Devops**, les équipes peuvent offrir en permanence du code sûr, testé et prêt à être utilisé en production. Cela inclut la publication de mises à jour logicielles, ce qui, dans une entreprise de télécommunication, peut parfois survenir trois fois par jour, voire plus.

1.2.3 Définition 3, extrait p.53 MISC N95, Janvier/février, 2018, "Ne pas prévoir, c'est déjà gémir"

L'ère des hyperviseurs est-elle révolue ? La bataille commerciale autour de la sécurité et de la performance persiste-t-elle ?

C'est à présent un conflit dépassé, car la sécurité est prise en compte désormais dans les conteneurs au niveau des prérequis.

L'importance du choix de la sécurité réside davantage dans l'édifice construit et son évolution.

Il devient évident que la **virtualisation légère** va gagner du terrain, les hyperviseurs vont alors devenir obsolètes et c'est dans ce contexte qu'il faut repenser l'action des équipes de sécurité.

En faisant avancer les vrais échanges entre Dev et Ops, le **Devops** a changé la donne et la **production bénéficie enfin de l'agilité prônée depuis quelques années**.

En intégrant la sécurité dans le **SecDevops**, et en s'assurant d'avoir des composants sécurisés au maximum, l'aspect sécuritaire devient alors une composante à valeur ajoutée pour la production.

Certains pensent qu'utiliser les systèmes qui ont fait leur preuve dans le temps serait le gage d'une sécurité beaucoup plus fiable et plus simple à mettre en œuvre.

Il semble aujourd'hui de plus en plus évident pour un responsable de systèmes d'information que manquer ce tournant de la technologie des conteneurs, serait une **assurance d'être rapidement mis à l'écart des évolutions en cours**.

1.2.3.1 Citations

1.2.3.1.1 Ne pas prévoir, c'est déjà gémir

“Ne pas prévoir, c'est déjà gémir” Léonard de Vinci.

1.2.3.1.2 La vie, c'est comme une bicyclette, il faut avancer pour ne pas perdre l'équilibre

La vie, c'est comme une bicyclette, il faut avancer pour ne pas perdre l'équilibre Albert Einstein

1.2.4 Devops, intégration et déploiement continu, pourquoi est-ce capital et comment y aller ?

See also:

- <https://actu.alfa-safety.fr/devops/devops-integration-et-deploiement-continus-pourquoi-est-ce-capital-et-comment-y-aller/>

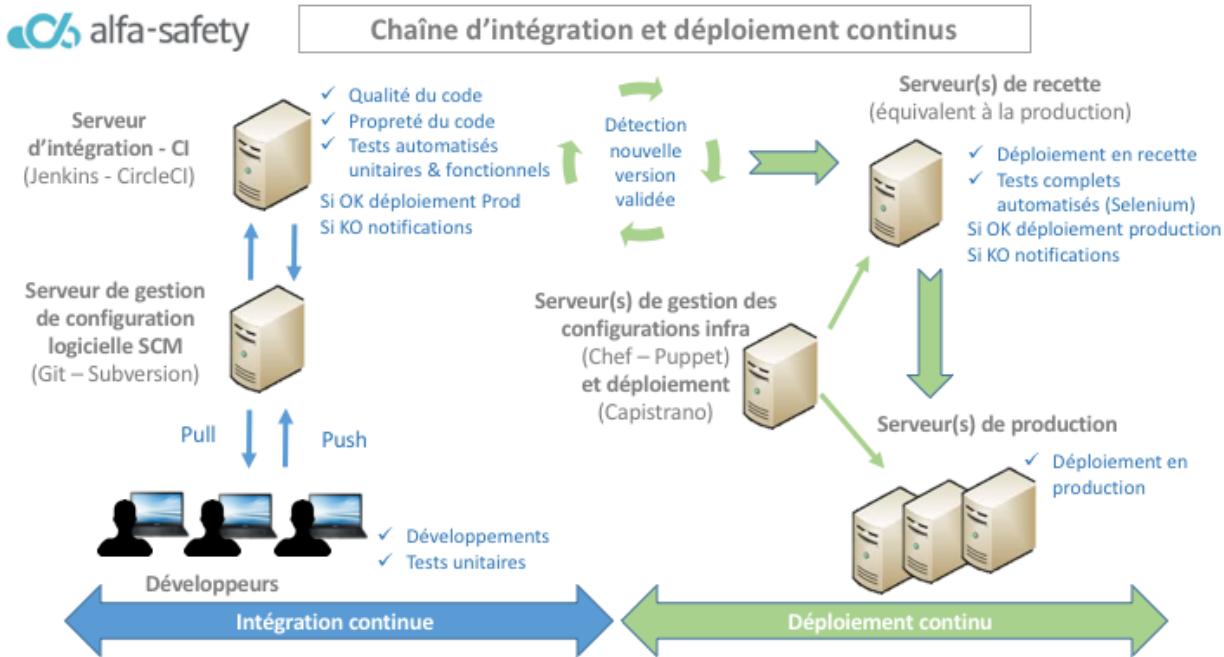


Fig. 2: Intégration continue

« Intégration continue » (CI), « déploiement continu » (CD), « Devops », autant de termes que l'on entend très fréquemment dès que l'on parle d'applications Web et de transformation numérique, et pourtant ce sont des concepts encore mal connus dans leur mise en œuvre.

De quoi s'agit-il ? Tout simplement d'assurer la sortie de nouvelles fonctionnalités d'une application sur un rythme beaucoup plus régulier et rapide.

Traditionnellement, un rythme de déploiement standard sur une application classique est d'une à deux versions majeures par an. Pour chaque version majeure, on regroupe un ensemble de nouvelles fonctionnalités, ce qui donne délai de 6 à 12 mois entre deux nouveautés.

Entretemps, on se contente de corriger les bugs, de sortir des versions mineures. C'est terriblement long, surtout à l'ère d'internet. L'objectif est d'assurer la cohérence des évolutions, regrouper les testss, sécuriser la production et limiter les migrations pour les clients, mais cela pénalise les délais.

Ce délai s'explique par le fait que c'est un processus séquentiel, impliquant différentes équipes et qu'à chaque étape, il faut synchroniser les acteurs, faire des demandes, les planifier, tout cela générant des délais.

Le déploiement continu prend le contrepied et permet d'accélérer ce rythme en :

- découplant les versions en un plus grand nombre de livraisons de moindre taille et moins complexes à tester,
- automatisant au maximum les étapes de tests et passages en production d'une nouvelle version afin de réduire les cycles,
- permettant un déploiement très régulier des nouveautés.

1.2.5 Agilité et Devops: Extrait p. 35 de [Programmez!] , N°214, janvier 2018

See also:

- <https://www.programmez.com/magazine/article/agilite-developpeurs/Devops-une-bonne-idee>

Les développeurs **doivent** évoluer pour suivre ces deux mouvements populaires (Agilité + **Devops**) qui se déplient très rapidement au sein de l'ensemble des DSI françaises. L'agilité et le **Devops** sont de très bonnes évolutions tant elles apportent aux DSI et au produit final.

1.2.6 What is a DevOps Engineer ?

See also:

<http://blog.shippable.com/how-to-be-a-great-devops-engineer>

A major part of adopting DevOps is to create a better working relationship between development and operations teams.

Some suggestions to do this include seating the teams together, involving them in each other's processes and workflows, and even creating one cross-functional team that does everything.

In all these methods, Dev is still Dev and Ops is still Ops.

The term DevOps Engineer tries to blur this divide between Dev and Ops altogether and suggests that the best approach is to hire engineers who can be excellent coders as well as handle all the Ops functions.

In short, a DevOps engineer can be a developer who can think with an Operations mindset and has the following skillset:

- Familiarity and experience with a variety of Ops and Automation tools
- Great at writing scripts
- Comfortable with dealing with frequent testing and incremental releases
- Understanding of Ops challenges and how they can be addressed during design and development
- Soft skills for better collaboration across the team

According to Amazon CTO Werner Vogels:

Giving developers operational responsibilities has greatly enhanced the quality of the services, both **from a** customer **and** a technology point of view.

The traditional model **is** that you take your software to the wall

(continues on next page)

(continued from previous page)

that separates development **and** operations, **and** throw it over **and** then forget about it. Not at Amazon. You build it, you run it. This brings developers into contact **with** the day-to-day operation of their software. It also brings them into day-to-day contact **with** the customer. This customer feedback loop **is** essential **for** improving the quality of the service.

It is easier than ever before for a developer to move to a DevOps role. Software delivery automation is getting better every day and DevOps platforms like Shippable are making it easy to implement automation while also giving you a Single Pane of Glass view across your entire CI/CD pipeline.

Can an Ops engineer move to a DevOps role? Definitely, but it can be a little more challenging since you will need to learn design and programming skills before making that transformation. However, with the upsurge in number of coding bootcamps, it is probably an easier transition to make than it was a few years ago. Ops engineers can bring much needed insights into how software design can cause Ops challenges, so once you get past the initial learning curve for design/coding, you're likely to become a valued DevOps engineer.

1.3 Définitions concernant Docker

See also:

- <https://www.docker.com/what-docker>

1.3.1 Définition de Docker sur Wikipedia en français

Docker est un logiciel libre qui automatise le déploiement d'applications dans des conteneurs logiciels. Selon la firme de recherche sur l'industrie 451 Research:

Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur.

Ceci permet d'étendre la flexibilité et la portabilité d'exécution d'une application, que ce soit sur la machine locale, un cloud privé ou public, une machine nue, etc

1.3.2 Docker est “agile”

Améliorations des temps de développement et de déploiement par 13.

1.3.3 Docker est portable

Docker est portable ce qui permet d'avoir des environnements de développement, test et production pratiquement identiques.

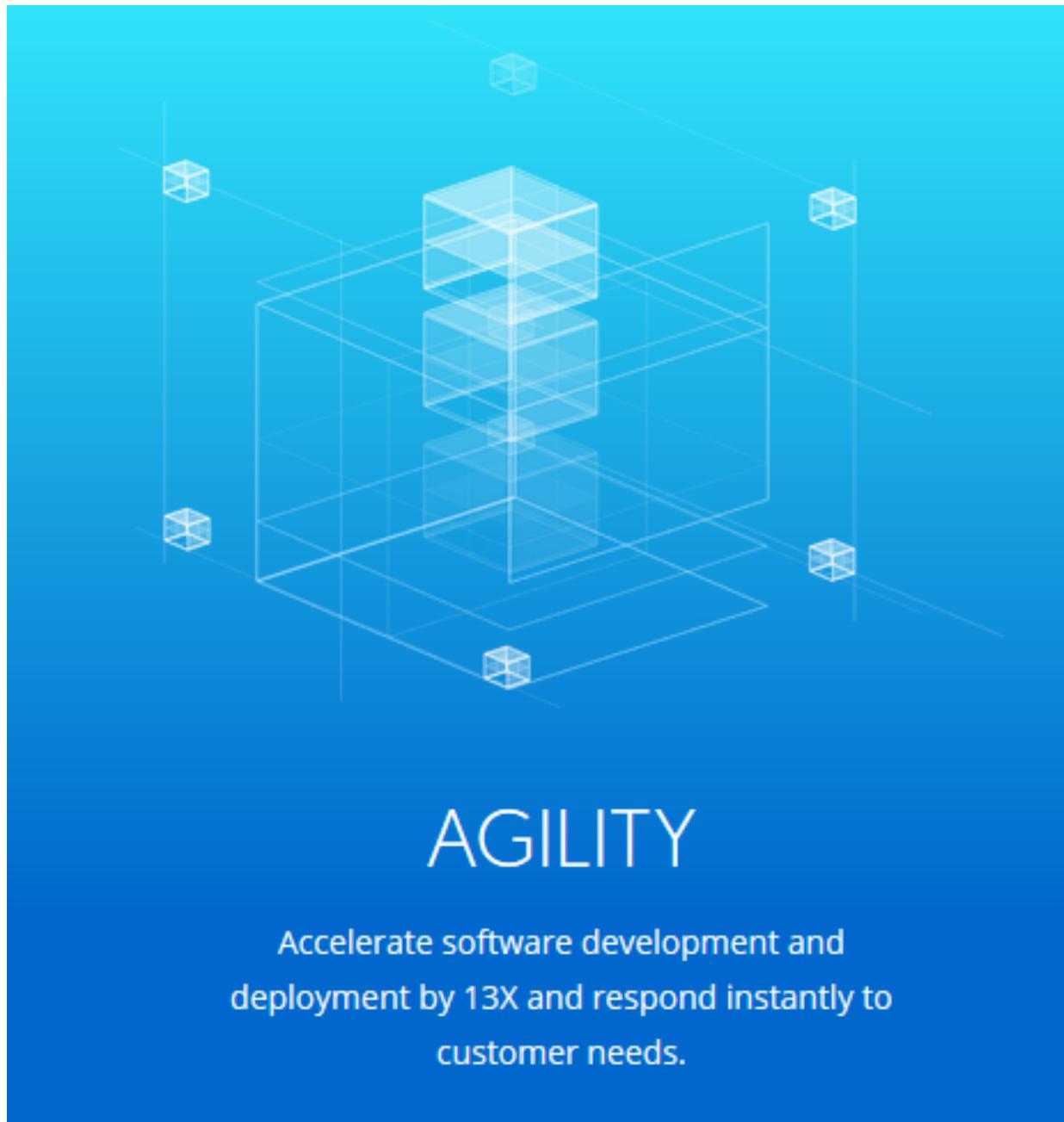


Fig. 3: Source: <https://www.docker.com/what-docker>

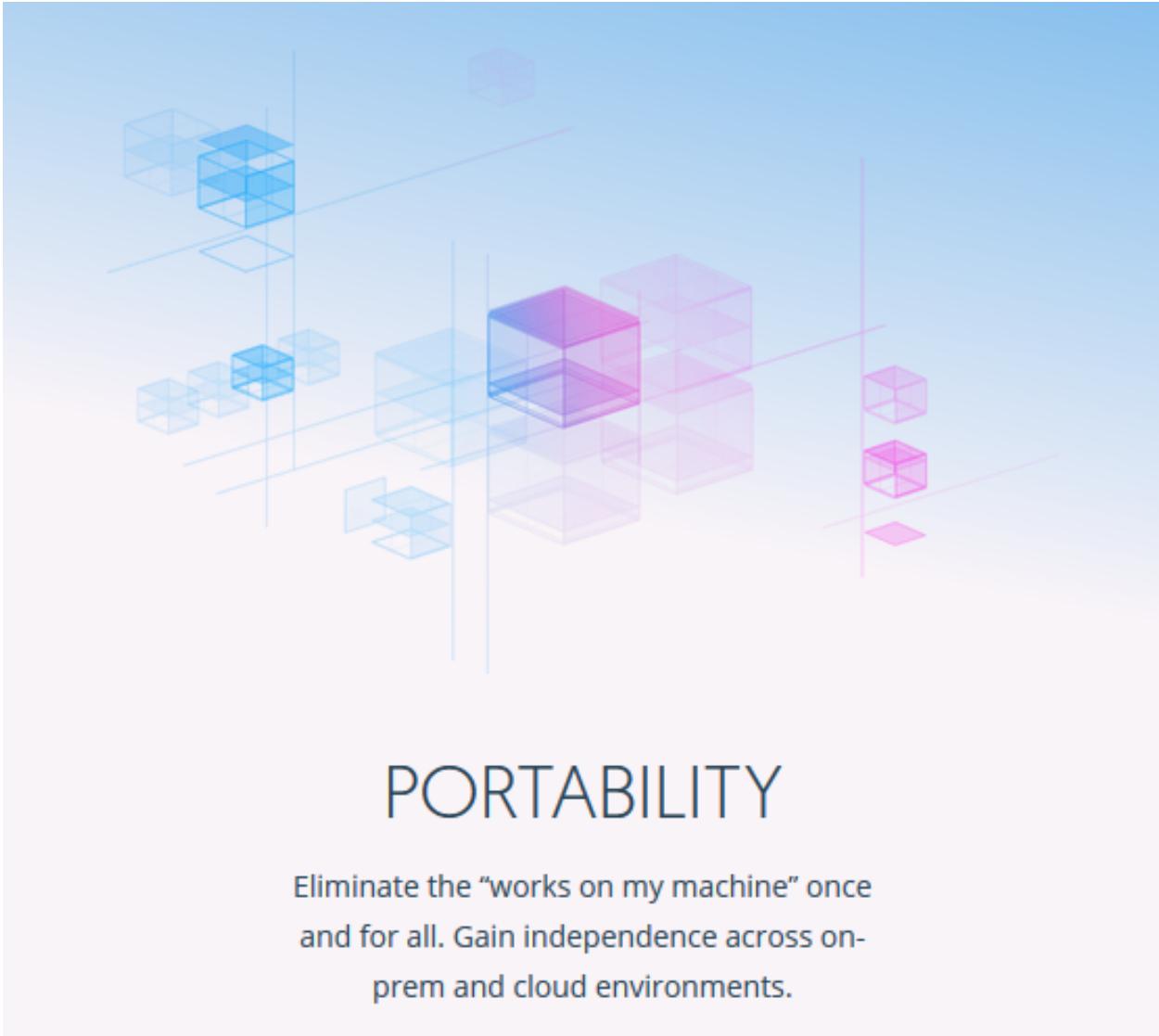


Fig. 4: Source: <https://www.docker.com/what-docker>

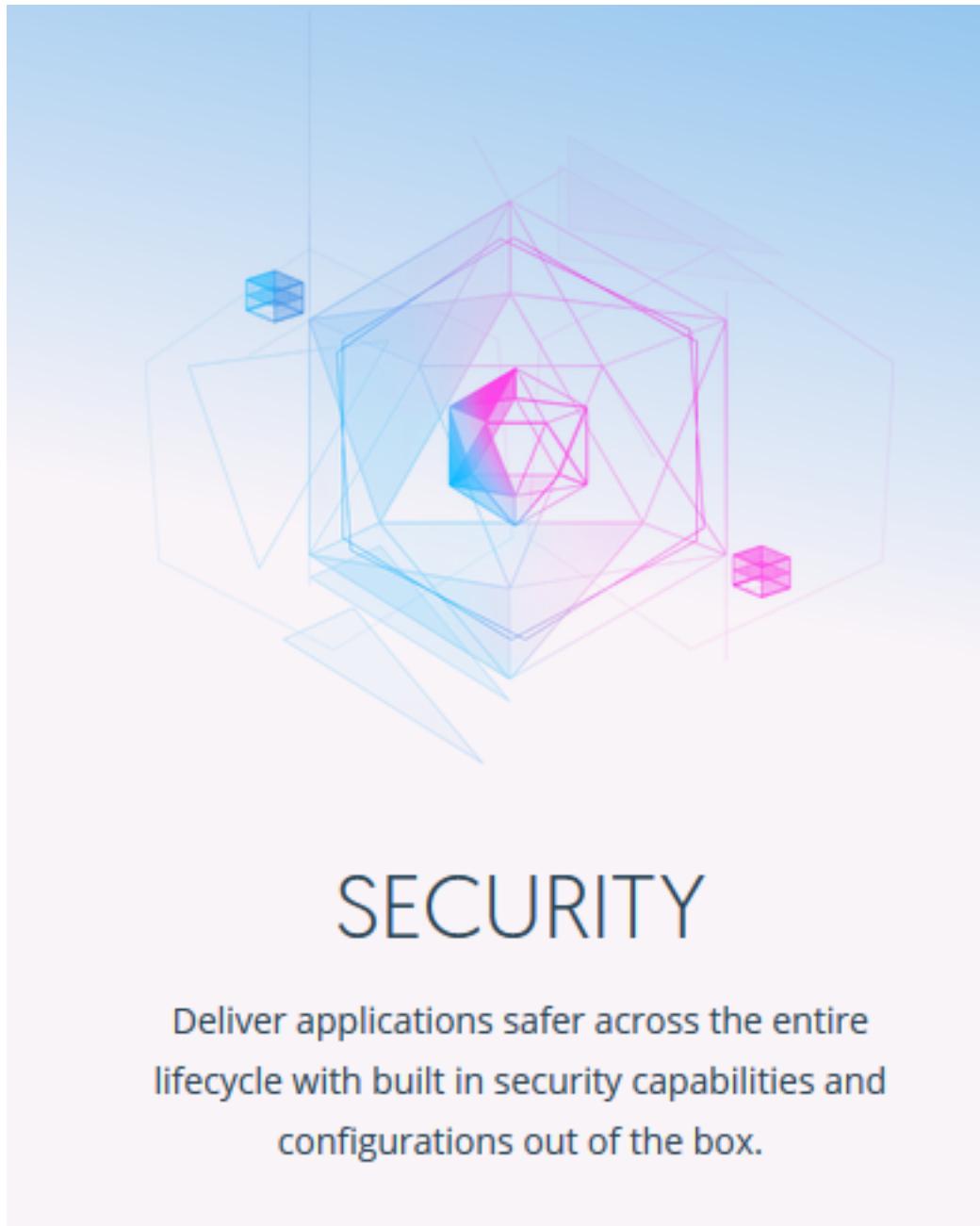


Fig. 5: Source: <https://www.docker.com/what-docker>

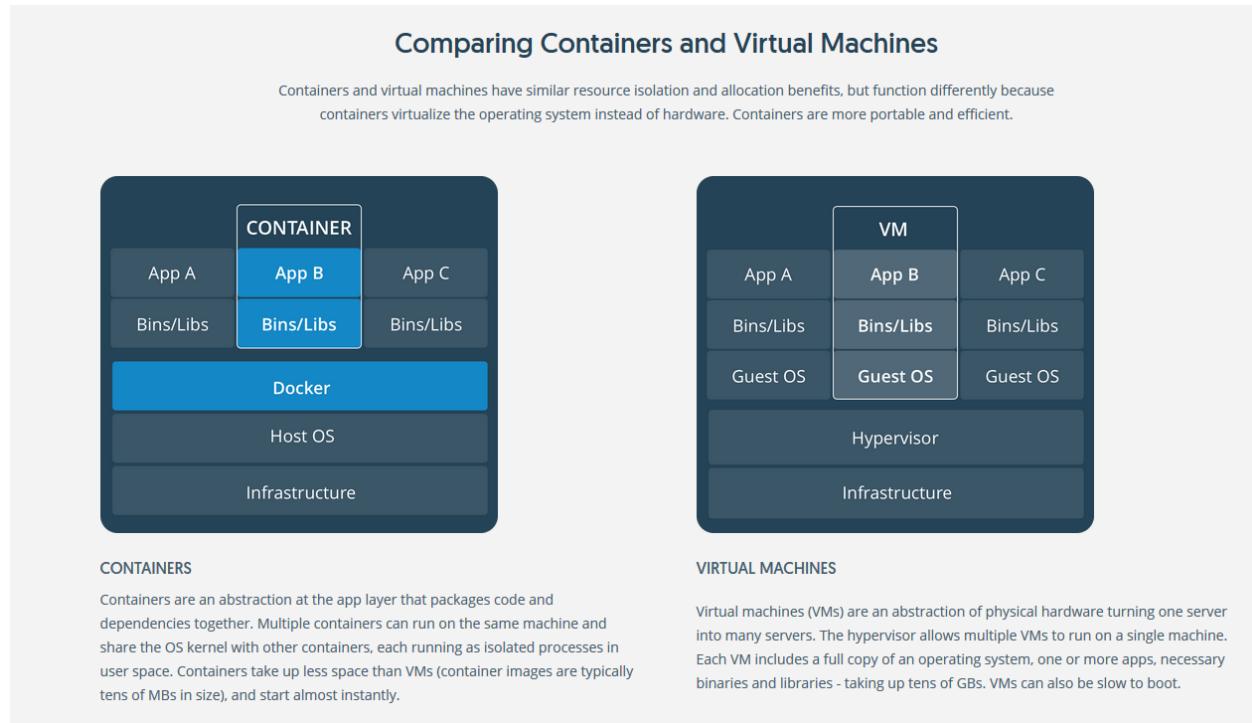


Fig. 6: Source: <https://www.docker.com/what-container>

1.3.4 Docker est sécurisé

1.3.5 Les conteneurs Docker sont plus légers et rapides que les machines virtuelles

1.3.5.1 Conteneurs

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.

Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

1.3.5.2 Virtual machines (VMs)

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers.

The hypervisor allows multiple VMs to run on a single machine.

Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs.

VMs can also be slow to boot.

1.3.5.3 Docker can run your applications in production at native speed

Source: p.255 du livre “Python Microservices Development” de Tarek Ziadé.

```
...  
that is where VMs are a great solution to run your applications.  
...
```

In the past ten years, many software projects that required an elaborate setup to run started to provide read-to-run VMs, using tools such as VMWare or VirtualBox. Those VMs included the whole stack, like prefilled databases. Demos became easily runnable on most platforms with a single command. That was progress.

However, some of those tools were not fully open source virtualization tool and they were very slow to run, and greedy in memory and CPU and terrible with disk I/O. It was **unthinkable** to run them in production, and they were mostly used for demos.

The **big revolution** came with Docker, an open source virtualization tool, which was first released in 2013, and became hugely popular. Moreover, unlike VMWare or VirtualBox, Docker can run your applications in production at **native speed**.

1.4 Dossier Docker dans le dossier MISC N°95 de janvier/février 2018

See also:

- <https://www.miscmag.com/misc-n95-references-de-larticle-docker-les-bons-reflexes-a-adopter/>

CHAPTER 2

Installation de docker

See also:

- <https://docs.docker.com/install>

CHAPTER 3

Qui utilise Docker en production ?

See also:

- <https://actu.alfa-safety.fr/devops/docker-en-production/>

Contents

- *Qui utilise Docker en production ?*
 - *Historique*
 - * *Janvier 2018*
 - *Paypal*
 - * *Challenges*
 - * *Solution*

3.1 Historique

3.1.1 Janvier 2018

See also:

- <https://blog.docker.com/2018/01/5-tips-learn-docker-2018/>

As the holiday season ends, many of us are making New Year's resolutions for 2018. Now is a great time to think about the new skills or technologies you'd like to learn. So much can change each year as technology progresses and companies are looking to innovate or modernize their legacy applications or infrastructure.

At the same time the market for Docker jobs continues to grow as companies such as Visa, MetLife and Splunk adopt Docker Enterprise Edition (EE) in production.

3.2 Paypal

See also:

- <https://www.docker.com/customers/paypal-uses-docker-containerize-existing-apps-save-money-and-boost-security>

3.2.1 Challenges

Today PayPal is leveraging OpenStack for their private cloud and runs more than 100,000 VMs. This private cloud runs 100% of their web and mid-tier applications and services. One of the biggest desires of the PayPal business is to modernize their datacenter infrastructure, making it more on demand, improving its security, meeting compliance regulations and lastly, making everything cost efficient.

They wanted to refactor their existing Java and C++ legacy applications by dockerizing them and deploying them as containers.

This called for a technology that provides a distributed application deployment architecture and can manage workloads but must also be deployed in both private, and eventually public cloud environments. Being cost efficient was extremely important for the company. Since PayPal runs their own cloud, they pay close attention to how much money they are spending on actually running their datacenter infrastructure.

Functioning within the online payment industry, PayPal must ensure the security of their internal data (binaries and artifacts with the source code of their applications). This makes them a **very security-conscious company**.

Their sensitive data needs to be kept on-premises where their security teams can run ongoing scans and sign their code before deploying out to production. PayPal's massive popularity is a good thing, but it also means they must handle the deluge of demands from their users. At times they process more than 200 payments per second. When including Braintree and Venmo, the companies that PayPal acquired, that number continues to soar even higher. Recently, it was announced that Braintree is processing more than a billion a month when it comes to mobile payments!. That adds quite a bit of extra pressure on their infrastructure.

3.2.2 Solution

Today PayPal uses Docker's commercial solutions to enable them to not only provide gains for their developers, in terms of productivity and agility, but also for their infrastructure teams in the form of cost efficiency and enterprise-grade security.

The tools being used in production today include:

- Docker Commercially Supported engine (CS Engine),
- Docker Trusted Registry
- as well as Docker Compose.

The company believes that containers and VMs can coexist and combine the two technologies. Leveraging Docker containers and VMs together gives PayPal the ability to run more applications while reducing the number of total VMs, optimizing their infrastructure. This also allows PayPal to spin up a new application much more quickly, and on an “as needed” basis.

Since containers are more lightweight and instantiate in a fraction of a second, while VMs take minutes, they can roll out a new application instance quickly, patch an existing application, or even add capacity for holiday readiness to compensate for peak times within the year.

This helps drive innovation and help them outpace competition. Docker Trusted Registry gives their team enterprise security features like granular role based access controls, and image signing that ensures that all of PayPal's checks and balances are in place.

The tool provides them with the on-premises enterprise-grade registry service they need in order to provide secure collaboration for their image content. Their security team can run ongoing scans and sign code before deploying to production.

With Docker, the company has gained the ability to scale quickly, deploy faster, and one day even provide local desktop-based development environments with Docker. For that, they are looking to Docker for Mac and Docker for Windows, which offer Docker as a local development environment to their 4,000+ developers located across the globe.

CHAPTER 4

docker engine CE (Community Edition)

See also:

- <https://github.com/docker/docker-ce>
- <https://docs.docker.com/engine/deprecated/>

Contents

- *docker engine CE (Community Edition)*
 - *docker engine versions*

4.1 docker engine versions

4.1.1 docker engine versions

See also:

- <https://github.com/docker/docker-ce/releases>
- <https://github.com/docker/docker.github.io/tree/vnext-engine>
- <https://docs.docker.com/engine/deprecated/>

Contents

- *docker engine versions*
 - *Future*
 - *Versions*

4.1.1.1 Future

See also:

- <https://github.com/docker/docker.github.io/tree/vnext-engine>
- <https://github.com/docker/docker-ce/commits/master>

4.1.1.2 Versions

4.1.1.2.1 18.09-ce (2018-11-08)

See also:

- <https://github.com/docker/docker-ce/tree/v18.09.0>
- <https://github.com/docker/docker-ce/releases/tag/v18.09.0>
- <https://docs.docker.com/engine/release-notes/#1809>

4.1.1.2.2 18.06.1-ce (2018-08-21)

See also:

- <https://github.com/docker/docker-ce/tree/v18.06.1-ce>

4.1.1.2.3 18.06.0-ce (2018-07-18)

See also:

- <https://github.com/docker/docker-ce/tree/v18.06.0-ce>

4.1.1.2.4 18.03.1-ce (2018-04-26, 9ee9f40)

See also:

- <https://github.com/docker/docker-ce/tree/v18.03.1-ce>

4.1.1.2.5 17.12.1-ce (2018-02-27)

See also:

- <https://github.com/docker/docker-ce/tree/v17.12.1-ce>

4.1.1.2.6 17.06.0-ce (2017-06-23, 02c1d87)

See also:

- <https://github.com/docker/docker-ce/tree/v17.06.0-ce>

CHAPTER 5

Docker compose

See also:

- <https://github.com/docker/compose/>
- <https://docs.docker.com/compose/>
- <https://docs.docker.com/compose/samples-for-compose/>
- <https://www.aquasec.com/wiki/display/containers/Docker+Compose>

Contents

- *Docker compose*
 - *Concepts clés*
 - * *Other definition*
 - *Other links*
 - * *heroku*
 - *docker-compose commands*
 - *docker-compose for production*
 - *docker-compose TIPS*
 - *docker-compose versions*
 - *Exemples*

5.1 Concepts clés

Key concepts these samples cover

The samples should help you to:

- define services based on Docker images using Compose files docker-compose.yml and docker-stack.yml files
- understand the relationship between docker-compose.yml and Dockerfiles
- learn how to make calls to your application services from Compose files
- learn how to deploy applications and services to a swarm

5.1.1 Other definition

Source: <https://blog.sixeyed.com/windows-dockerfile-26/>

You define applications in Compose in terms of services rather than containers. Services will be deployed as containers - but a service could be multiple instances of a single container, so it's an abstraction beyond managing containers on hosts.

5.2 Other links

See also:

- <https://www.aquasec.com/wiki/display/containers/Docker+Compose>

5.2.1 heroku

See also:

- <https://devcenter.heroku.com/articles/local-development-with-docker-compose>

5.3 docker-compose commands

5.3.1 docker-compose commands

See also:

- <https://docs.docker.com/compose/overview>
- <https://docs.docker.com/compose/reference>
- <https://docs.docker.com/compose/compose-file/>
- <https://docs.docker.com/compose/samples-for-compose>

Contents

- *docker-compose commands*
 - *docker-compose help*
 - *docker-compose build*
 - * *docker-compose -f docker-compose.yml build django*
 - *docker-compose ps*

- *docker-compose up*

5.3.1.1 docker-compose help

`docker-compose help`

Define **and** run multi-container applications **with** Docker.

Usage:

```
docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]
docker-compose -h|--help
```

Options:

<code>-f, --file FILE</code>	Specify an alternate compose file (default: <code>docker-compose.yml</code>)
<code>-p, --project-name NAME</code>	Specify an alternate project name (default: directory name)
<code>--verbose</code>	Show more output
<code>--log-level LEVEL</code>	Set log level (DEBUG, INFO, WARNING, ERROR, CRITICAL)
<code>--no-ansi</code>	Do not print ANSI control characters
<code>-v, --version</code>	Print version and exit
<code>-H, --host HOST</code>	Daemon socket to connect to
<code>--tls</code>	Use TLS; implied by <code>--tlsverify</code>
<code>--tlscacert CA_PATH</code>	Trust certs signed only by this CA
<code>--tlscert CLIENT_CERT_PATH</code>	Path to TLS certificate file
<code>--tlskey TLS_KEY_PATH</code>	Path to TLS key file
<code>--tlsverify</code>	Use TLS and verify the remote
<code>--skip-hostname-check</code>	Don't check the daemon's hostname against the name specified in the client certificate
<code>--project-directory PATH</code>	Specify an alternate working directory (default: the path of the Compose file)
<code>--compatibility</code>	If set , Compose will attempt to convert deploy keys in v3 files to their non-Swarm equivalent

Commands:

<code>build</code>	Build or rebuild services
<code>bundle</code>	Generate a Docker bundle from the Compose file
<code>config</code>	Validate and view the Compose file
<code>create</code>	Create services
<code>down</code>	Stop and remove containers, networks, images, and volumes
<code>events</code>	Receive real time events from containers
<code>exec</code>	Execute a command in a running container
<code>help</code>	Get help on a command
<code>images</code>	List images
<code>kill</code>	Kill containers
<code>logs</code>	View output from containers
<code>pause</code>	Pause services
<code>port</code>	Print the public port for a port binding
<code>ps</code>	List containers
<code>pull</code>	Pull service images
<code>push</code>	Push service images
<code>restart</code>	Restart services
<code>rm</code>	Remove stopped containers
<code>run</code>	Run a one-off command

(continues on next page)

(continued from previous page)

scale	Set number of containers for a service
start	Start services
stop	Stop services
top	Display the running processes
unpause	Unpause services
up	Create and start containers
version	Show the Docker-Compose version information

5.3.1.2 docker-compose build

See also:

- <https://docs.docker.com/compose/reference/build>

```
$ docker-compose help build
```

Build or rebuild services.

Services are built once and then tagged as `project_service`, e.g. `composetest_db`. If you change a service's `Dockerfile` or the contents of its build directory, you can run `docker-compose build` to rebuild it.

Usage: build [options] [--build-arg key=val...] [SERVICE...]

Options:

--compress	Compress the build context using gzip.
--force-rm	Always remove intermediate containers.
--no-cache	Do not use cache when building the image.
--pull	Always attempt to pull a newer version of the image.
-m, --memory MEM	Sets memory limit for the build container.
--build-arg key=val	Set build-time variables for services.

5.3.1.2.1 docker-compose -f docker-compose.yml build django

```
docker-compose -f docker-compose.yml build django
```

5.3.1.3 docker-compose ps

See also:

- <https://docs.docker.com/compose/reference/up>
- *Check container status*

```
$ docker-compose help ps
```

List containers.

Usage: ps [options] [SERVICE...]

Options:

-q, --quiet	Only display IDs
-------------	------------------

(continues on next page)

(continued from previous page)

--services	Display services
--filter KEY=VAL	Filter services by a property

5.3.1.4 docker-compose up

See also:

- <https://docs.docker.com/compose/reference/up>
- *Launching Our First Stack with Compose*

```
docker-compose help up
```

Builds, (re)creates, starts, and attaches to containers for a service.

Unless they are already running, this command also starts any linked services.

The `docker-compose up` command aggregates the output of each container. When the command exits, all containers are stopped. Running `docker-compose up -d` starts the containers in the background and leaves them running.

If there are existing containers for a service, and the service's configuration or image was changed after the container's creation, `docker-compose up` picks up the changes by stopping and recreating the containers (preserving mounted volumes). To prevent Compose from picking up changes, use the `--no-recreate` flag.

If you want to force Compose to stop and recreate all containers, use the `--force-recreate` flag.

Usage: `up [options] [--scale SERVICE=NUM...] [SERVICE...]`

Options:

-d, --detach	Detached mode: Run containers in the background, print new container names. Incompatible with --abort-on-container-exit.
--no-color	Produce monochrome output.
--quiet-pull	Pull without printing progress information
--no-deps	Don't start linked services.
--force-recreate	Recreate containers even if their configuration and image haven't changed.
--always-recreate-deps	Recreate dependent containers. Incompatible with --no-recreate.
--no-recreate	If containers already exist, don't recreate them. Incompatible with --force-recreate and -V.
--no-build	Don't build an image, even if it's missing.
--no-start	Don't start the services after creating them.
--build	Build images before starting containers.
--abort-on-container-exit	Stops all containers if any container was stopped. Incompatible with -d.
-t, --timeout TIMEOUT	Use this timeout in seconds for container shutdown when attached or when containers are already running. (default: 10)
-V, --renew-anon-volumes	Recreate anonymous volumes instead of retrieving data from the previous containers.

(continues on next page)

(continued from previous page)

--remove-orphans	Remove containers for services not defined in the Compose file.
--exit-code-from SERVICE	Return the exit code of the selected service container. Implies --abort-on-container-exit.
--scale SERVICE=NUM	Scale SERVICE to NUM instances. Overrides the `scale` setting in the Compose file if present.

5.4 docker-compose for production

5.4.1 Docker compose in production

See also:

- <https://docs.docker.com/compose/production/>
- <https://docs.docker.com/compose/production/#modify-your-compose-file-for-production>

Contents

- *Docker compose in production*
 - *Articles*
 - * *Simple continuous deployment with docker compose, docker machine and Gitlab CI (2017-06-26)*

5.4.1.1 Articles

5.4.1.1.1 Simple continuous deployment with docker compose, docker machine and Gitlab CI (2017-06-26)

See also:

- <https://medium.com/@Empanado/simple-continuous-deployment-with-docker-compose-docker-machine-and-gitlab-ci-9047765>

For local development of microservice-based systems running on docker, we've found that docker compose is probably the best way to go for local development, with the docker compose yaml file format being very usable for configuration as well. And for some projects, there really is no need to scale up to having multiple containers of a service, as you'll be just fine with running all your containers on a single host. For these projects you want to get to production as smooth (or simple) as possible.

So after spending time learning about Mesos, Kubernetes, Amazon ECS and other proprietary technologies and learning a ton of new concepts, I concluded that they're all awesome, but not really suitable for a simple move from local development with docker compose. They all have their own configuration formats (largely for good reasons) and all of them do orchestration quite a bit different than docker compose, to facilitate more complex deployment environments.

5.5 docker-compose TIPS

5.5.1 docker-compose tips

5.5.1.1 docker-compose tips 2018

Contents

- *docker-compose tips 2018*
 - *3 Docker Compose features for improving team development workflow*
 - * *Environment variables*
 - * *Templating*
 - * *Control your Compose Command Scope*

5.5.1.1.1 3 Docker Compose features for improving team development workflow

See also:

- <https://www.oreilly.com/ideas/3-docker-compose-features-for-improving-team-development-workflow>
- <https://twitter.com/BretFisher>

Environment variables

See also:

- <https://docs.docker.com/compose/compose-file/#variable-substitution>

Eventually, you'll need a compose file to be flexible and you'll learn that you can use environment variables inside the Compose file.

Note, this is not related to the YAML object “environment,” which you want to send to the container on startup. With the notation of \${VARNAME}, you can have Compose resolve these values dynamically during the processing of that YAML file. The most common examples of when to use this are for setting the container image tag or published port.

As an example, if your docker-compose.yml file looks like this

```
version: '2'
services:
  ghost:
    image: ghost:${GHOST_VERSION}
```

then you can control the image version used from the CLI like so:

```
GHOST_VERSION=2 docker-compose up
```

You can also set those variables in other ways: by storing them in a .env file, by setting them at the CLI with export, or even setting a default in the YAML itself with \${GHOST_VERSION:-2}.

You can read more about variable substitution and various ways to set them in the Docker docs.

Templating

See also:

- <https://docs.docker.com/compose/compose-file/#extension-fields>

A relatively new and lesser-known feature is Extension Fields, which lets you define a block of text in Compose files that is reused throughout the file itself.

This is mostly used when you need to set the same environment objects for a bunch of microservices, and you want to keep the file DRY (Don't Repeat Yourself).

I recently used it to set all the same logging options for each service in a Compose file like so:

```
version: '3.4'

x-logging:
  &my-logging
  options:
    max-size: '1m'
    max-file: '5'

  services:
    ghost:
      image: ghost
      logging: *my-logging
    nginx:
      image: nginx
      logging: *my-logging
```

You'll notice a new section starting with an x-, which is the template, that you can then name with the & and call from anywhere in your Compose file with * and the name. Once you start to use microservices and have hundreds or more lines in your Compose file, this will likely save you considerable time and ensure consistency of options throughout.

See more details in the Docker docs

Control your Compose Command Scope

The docker-compose CLI controls one or more containers, volumes, networks, etc., within its scope.

It uses two things to create that scope: the Compose YAML config file (it defaults to docker-compose.yml) and the project name (it defaults to the directory name holding the YAML config file). Normally you would start a project with a single docker-compose.yml file and execute commands like docker-compose up in the directory with that file, but there's a lot of flexibility here as complexity grows.

As things get more complex, you may have multiple YAML config files for different setups and want to control which one the CLI uses, like docker-compose -f custom-compose.yml up. This command ignores the default YAML file and only uses the one you specify with the -f option.

You can combine many Compose files in a layered override approach. Each one listed in the CLI will override the settings of the previous (processed left to right):

```
docker-compose -f docker-compose.yml -f docker-override.yml
```

If you manually change the project name, you can use the same Compose file in multiple scopes so they don't "clash." Clashing happens when Compose tries to control a container that already has another one running with the same name.

You likely have noticed that containers, networks, and other objects that Compose creates have a naming standard. The standard comprises three parts: `projectname_servicename_index`. We can change the `projectname`, which again, defaults to the directory name with a `-p` at the command line. So if we had a `docker-compose.yml` file like this:

```
version: '2'

services:
  ghost:
    image: ghost:${GHOST_VERSION}
    ports:
      - ${GHOST_PORT}:2368
```

Then we had it in a directory named “`app1`” and we started the `ghost` app with inline environment variables like this:

```
app1> GHOST_VERSION=2 GHOST_PORT=8080 docker-compose up
```

We'd see a container running named this:

```
app1_ghost_1
```

Now, if we want to run an older version of `ghost` side-by-side at the same time, we could do that with this same Compose file, as long as we change two things.:

- First, we need to change the project name to ensure the container name will be different and not conflict with our first one.
- Second, we need to change the published port so they don't clash with any other running containers.

```
app1> GHOST_VERSION=1 GHOST_PORT=9090 docker-compose -p app2 up
```

If I check running containers with a `docker container ls`, I see:

```
app1_ghost_1 running ghost:2 on port 8080
app2_ghost_1 running ghost:1 on port 9090
```

Now you could pull up two browser windows and browse both 8080 and 9090 with two separate `ghost` versions (and databases) running side by side.

Most of what I've learned on advanced Compose workflows has come from trying things I've learned in the Docker docs, as well as the teams I work with to make development, testing, and deployments easier.

I share these learnings everywhere I can, and I encourage you to do the same.

What other features or team standards have you found useful with Docker Compose? Please share with me and the community on [Twitter @BretFisher](#).

5.6 docker-compose versions

5.6.1 docker-compose versions

Contents

- *docker-compose versions*
 - *Versions*

5.6.1.1 Versions

5.6.1.1.1 docker-compose 1.22 (2018-07-18)

Contents

- *docker-compose 1.22 (2018-07-18)*
 - *docker-compose 1.22 (2018-07-18)*

docker-compose 1.22 (2018-07-18)

See also:

- <https://github.com/docker/compose/tree/1.22.0>

5.6.1.1.2 docker-compose 1.21 (2018-04-09)

Contents

- *docker-compose 1.21 (2018-04-09)*
 - *docker-compose 1.21.2 (2018-05-02)*

docker-compose 1.21.2 (2018-05-02)

See also:

- <https://github.com/docker/compose/tree/1.21.2>

```
$ docker-compose version
```

```
docker-compose version 1.21.2, build a133471
docker-py version: 3.3.0
CPython version: 3.6.5
OpenSSL version: OpenSSL 1.0.1t  3 May 2016
```

5.7 Exemples

5.7.1 Quickstart: Compose and Django

See also:

- <https://docs.docker.com/compose/django/>
- <https://docs.docker.com/compose/install/>
- <https://docs.docker.com/engine/tutorials/dockerimages/#building-an-image-from-a-dockerfile>

- <https://docs.docker.com/engine/reference/builder/>
- <https://store.docker.com/images/python>
- <https://docs.docker.com/compose/compose-file/>
- <https://docs.djangoproject.com/en/1.11/ref/settings/#allowed-hosts>
- <https://docs.docker.com/compose/reference/down/>

Contents

- *Quickstart: Compose and Django*
 - *Overview of Docker Compose*
 - *Introduction*
 - *Define the project components*
 - * *mkdir django_app*
 - * *Create a Dockerfile*
 - *Les images Python*
 - * *Create a requirements.txt in your project directory*
 - * *Create a file called docker-compose.yml in your project directory*
 - *Les images postgresql*
 - *Create a Django project*
 - * *cd django_app*
 - * *docker-compose run web django-admin.py startproject composeexample .*
 - *tree /a /f.*
 - *Connect the database*
 - * *Edit the composeexample/settings.py file*
 - * *django_app> docker-compose up*
 - * *docker ps*
 - * *docker-compose down*
 - *Compose file examples*

5.7.1.1 Overview of Docker Compose

See also:

- <https://github.com/docker/compose>
- <https://github.com/docker/docker.github.io/blob/master/compose/django.md>
- <https://docs.docker.com/compose/overview/>
- <https://docs.docker.com/compose/compose-file/>
- <https://github.com/docker/docker.github.io/blob/master/compose/overview.md#common-use-cases>

Looking for Compose file reference? Find the latest version [here](#).

Compose is a tool for defining and running multi-container Docker applications.

With Compose, you use a YAML file to configure your **application's services**.

Then, with a single command, you create and start all the services from your configuration. To learn more about all the features of Compose, see the [list of features](#).

Compose works in all environments:

- production,
- staging,
- development,
- testing,
- as well as CI workflows.

You can learn more about each case in [Common Use Cases](#).

Using Compose is basically a three-step process:

- Define your app's environment with a Dockerfile so it can be reproduced anywhere.
- Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
- Lastly, run **docker-compose up** and Compose will start and run your entire app.

5.7.1.2 Introduction

This quick-start guide demonstrates how to use Docker Compose to set up and run a simple Django/*PostgreSQL app*.

Before starting, you'll need to have Compose installed.

5.7.1.3 Define the project components

For this project, you need to create a Dockerfile, a Python dependencies file, and a docker-compose.yml file. (You can use either a .yml or .yaml extension for this file.)

5.7.1.3.1 mkdir django_app

Create an empty project directory.

You can name the directory something easy for you to remember. This directory is the context for your application image. The directory should only contain resources to build that image.

```
mkdir django_app
```

5.7.1.3.2 Create a Dockerfile

Create a new file called Dockerfile in your project directory.

```

1 FROM python:3
2 ENV PYTHONUNBUFFERED 1
3 RUN mkdir /code
4 WORKDIR /code
5 ADD requirements.txt /code/
6 RUN pip install -r requirements.txt
7 ADD . /code/

```

The Dockerfile defines an application's image content via one or more build commands that configure that image.

Once built, you can run the image in a container.

For more information on Dockerfile, see the [Docker user guide](#) and the [Dockerfile reference](#).

This Dockerfile starts with a Python 3 parent image.

Les images Python

Le tag **python:3** correspond à la version courante en 2018 c'est à dire **3.6**.

The parent image is modified by adding a new code directory. The parent image is further modified by installing the Python requirements defined in the requirements.txt file.

5.7.1.3.3 Create a requirements.txt in your project directory

This file is used by the RUN pip install -r requirements.txt command in your Dockerfile.

```

1 django
2 psycopg2

```

5.7.1.3.4 Create a file called docker-compose.yml in your project directory

See also:

- <https://docs.docker.com/compose/compose-file/>
- <https://store.docker.com/images/postgres>

The docker-compose.yml file describes the services that make your app.

```

1 version: '3'
2
3 services:
4   db:
5     image: postgres
6   web:
7     build: .
8     command: python3 manage.py runserver 0.0.0.0:8000
9     volumes:
10      - .:/code
11     ports:
12       - "8000:8000"
13     depends_on:
14       - db

```

This file defines two services: The **db** service and the **web** service.

Shared Tags

- 3.7.0a4 , 3.7-rc , rc :
 - [3.7.0a4-stretch \(3.7-rc/stretch/Dockerfile\)](#)
 - [3.7.0a4-windowsservercore-1tsc2016 \(3.7-rc/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [3.7.0a4-windowsservercore-1709 \(3.7-rc/windows/windowsservercore-1709/Dockerfile\)](#)
- 3.7.0a4-windowsservercore , 3.7-rc-windowsservercore , rc-windowsservercore :
 - [3.7.0a4-windowsservercore-1tsc2016 \(3.7-rc/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [3.7.0a4-windowsservercore-1709 \(3.7-rc/windows/windowsservercore-1709/Dockerfile\)](#)
- 3.6.4 , 3.6 , 3 , latest :
 - [3.6.4-jessie \(3.6/jessie/Dockerfile\)](#)
 - [3.6.4-windowsservercore-1tsc2016 \(3.6/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [3.6.4-windowsservercore-1709 \(3.6/windows/windowsservercore-1709/Dockerfile\)](#)
- 3.6.4-windowsservercore , 3.6-windowsservercore , 3-windowsservercore , windowsservercore :
 - [3.6.4-windowsservercore-1tsc2016 \(3.6/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [3.6.4-windowsservercore-1709 \(3.6/windows/windowsservercore-1709/Dockerfile\)](#)
- 3.5.4 , 3.5 :
 - [3.5.4-jessie \(3.5/jessie/Dockerfile\)](#)
 - [3.5.4-windowsservercore-1tsc2016 \(3.5/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [3.5.4-windowsservercore-1709 \(3.5/windows/windowsservercore-1709/Dockerfile\)](#)
- 3.5.4-windowsservercore , 3.5-windowsservercore :
 - [3.5.4-windowsservercore-1tsc2016 \(3.5/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [3.5.4-windowsservercore-1709 \(3.5/windows/windowsservercore-1709/Dockerfile\)](#)
- 3.4.7 , 3.4 :
 - [3.4.7-jessie \(3.4/jessie/Dockerfile\)](#)
- 2.7.14 , 2.7 , 2 :
 - [2.7.14-jessie \(2.7/jessie/Dockerfile\)](#)
 - [2.7.14-windowsservercore-1tsc2016 \(2.7/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [2.7.14-windowsservercore-1709 \(2.7/windows/windowsservercore-1709/Dockerfile\)](#)
- 2.7.14-windowsservercore , 2.7-windowsservercore , 2-windowsservercore :
 - [2.7.14-windowsservercore-1tsc2016 \(2.7/windows/windowsservercore-1tsc2016/Dockerfile\)](#)
 - [2.7.14-windowsservercore-1709 \(2.7/windows/windowsservercore-1709/Dockerfile\)](#)

Fig. 1: Les images Python voir <https://store.docker.com/images/python>

Les images postgresql

Supported tags and respective Dockerfile links

- [10.1 , 10 , latest \(10/Dockerfile\)](#)
- [10.1-alpine , 10-alpine , alpine \(10/alpine/Dockerfile\)](#)
- [9.6.6 , 9.6 , 9 \(9.6/Dockerfile\)](#)
- [9.6.6-alpine , 9.6-alpine , 9-alpine \(9.6/alpine/Dockerfile\)](#)
- [9.5.10 , 9.5 \(9.5/Dockerfile\)](#)
- [9.5.10-alpine , 9.5-alpine \(9.5/alpine/Dockerfile\)](#)
- [9.4.15 , 9.4 \(9.4/Dockerfile\)](#)
- [9.4.15-alpine , 9.4-alpine \(9.4/alpine/Dockerfile\)](#)
- [9.3.20 , 9.3 \(9.3/Dockerfile\)](#)
- [9.3.20-alpine , 9.3-alpine \(9.3/alpine/Dockerfile\)](#)

Fig. 2: Les images PostgreSQL voir <https://store.docker.com/images/postgres>

The compose file also describes which Docker images these services use, how they link together, any volumes they might need mounted inside the containers.

See the [docker-compose.yml reference](#) for more information on how this file works

5.7.1.4 Create a Django project

In this step, you create a Django starter project by building the image from the build context defined in the previous procedure.

5.7.1.4.1 cd django_app

Change to the root of your project directory.

5.7.1.4.2 docker-compose run web django-admin.py startproject composeexample .

This instructs Compose to run django-admin.py startproject composeexample in a container, using the web service's image and configuration.

Because the web image doesn't exist yet, Compose builds it from the current directory, as specified by the build: . line in docker-compose.yml.

```
docker-compose run web django-admin.py startproject composeexample .
```

```
Pulling db (postgres:latest)...
latest: Pulling from library/postgres
723254a2c089: Pull complete
39ec0e6c372c: Pull complete
ba1542fb91f3: Pull complete
c7195e642388: Pull complete
95424deca6a2: Pull complete
```

(continues on next page)

```
Y:\projects_id3\PSN001\XLOGCA135_tutorial_docker\tutorial_docker\compose\django\django_app>docker-compose run web django-admin.py startproject composeexample .
WARNING: The Docker Engine you're using is running in swarm mode.
Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To deploy your application across the swarm, use `docker stack deploy`.

Pulling db (postgres:latest)...
latest: Pulling from library/postgres
723254a2c089: Downloading [=====] 27.98MB/45.12MB
39ec0e6c372c: Download complete
ba1542fb91f3: Download complete
c7195e642388: Download complete
95424dec6a2: Download complete
2d7d4b3a4ce2: Download complete
fbde41d4a8cc: Download complete
880120b92add: Downloading [=====] 20.53MB/57.1MB
9a217c784089: Download complete
d581543fe8e7: Download complete
e5eff8940bb0: Download complete
462d60a56b09: Download complete
135fa6b9c139: Download complete

```

Fig. 3: docker-compose run web django-admin.py startproject composeexample .

(continued from previous page)

```
2d7d4b3a4ce2: Pull complete
fbde41d4a8cc: Pull complete
880120b92add: Pull complete
9a217c784089: Pull complete
d581543fe8e7: Pull complete
e5eff8940bb0: Pull complete
462d60a56b09: Pull complete
135fa6b9c139: Pull complete
Digest: sha256:3f4441460029e12905a5d447a3549ae2ac13323d045391b0cb0cf8b48ea17463
Status: Downloaded newer image for postgres:latest
Creating djangoapp_db_1 ... done
Building web
Step 1/7 : FROM python:3
3: Pulling from library/python
f49cf87b52c1: Already exists
7b491c575b06: Pull complete
b313b08bab3b: Pull complete
51d6678c3f0e: Pull complete
09f35bd58db2: Pull complete
0f9de702e222: Pull complete
73911d37fcde: Pull complete
99a87e214c92: Pull complete
Digest: sha256:98149ed5f37f48ea3fad26ae6c0042dd2b08228d58edc95ef0fce35f1b3d9e9f
Status: Downloaded newer image for python:3
--> cle459c00dc3
Step 2/7 : ENV PYTHONUNBUFFERED 1
--> Running in 94847219310a
Removing intermediate container 94847219310a
--> 221d2e9ab9e4
Step 3/7 : RUN mkdir /code
--> Running in a65c8bf5e5a9
Removing intermediate container a65c8bf5e5a9
--> 589950689c7a
Step 4/7 : WORKDIR /code
Removing intermediate container f7b978400775
--> e039064473fb
Step 5/7 : ADD requirements.txt /code/
--> 4305caf141b9
Step 6/7 : RUN pip install -r requirements.txt
--> Running in 0705839561d0
```

(continues on next page)

(continued from previous page)

```

Collecting django (from -r requirements.txt (line 1))
  Downloading Django-2.0.1-py3-none-any.whl (7.1MB)
Collecting psycopg2 (from -r requirements.txt (line 2))
  Downloading psycopg2-2.7.3.2-cp36-cp36m-manylinux1_x86_64.whl (2.7MB)
Collecting pytz (from django->-r requirements.txt (line 1))
  Downloading pytz-2017.3-py3-none-any.whl (511kB)
Installing collected packages: pytz, django, psycopg2
Successfully installed django-2.0.1 psycopg2-2.7.3.2 pytz-2017.3
Removing intermediate container 0705839561d0
--> fa8182703037
Step 7/7 : ADD . /code/
--> 72d70c82ea04
Successfully built 72d70c82ea04
Successfully tagged djangoapp_web:latest
WARNING: Image for service web was built because it did not already exist.
To rebuild this image you must use `docker-compose build` or `docker-compose up --build`.

```

Once the web service image is built, Compose runs it and executes the django-admin.py startproject command in the container. This command instructs Django to create a set of files and directories representing a Django project.

tree /a /f .

```
$ tree /a /f .
```

```

Y:\PROJECTS_ID3\P5N001\XLOGCA135_TUTORIAL_DOCKER\TUTORIAL_
└─DOCKER\COMPOSE\DJANGO\DJANGO_APP
|   docker-compose.yml
|   Dockerfile
|   manage.py
|   requirements.txt
|
└──composeexample
    settings.py
    urls.py
    wsgi.py
    __init__.py

```

5.7.1.5 Connect the database

See also:

- <https://store.docker.com/images/postgres>

In this section, you set up the database connection for Django.

5.7.1.5.1 Edit the composeexample/settings.py file

In your project directory, edit the composeexample/settings.py file.

Replace the DATABASES = ... with the following:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'HOST': 'db',
        'PORT': 5432,
    }
}
```

These settings are determined by the postgres Docker image specified in docker-compose.yml.

5.7.1.5.2 django_app> docker-compose up

Run the docker-compose up command from the top level directory for your project.

```
django_app>docker-compose up
```

```
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All ↴
containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

djangoapp_db_1 is up-to-date
Creating djangoapp_web_1 ... done
Attaching to djangoapp_db_1, djangoapp_web_1
db_1    | The files belonging to this database system will be owned by user "postgres".
db_1    | This user must also own the server process.
db_1    |
db_1    | The database cluster will be initialized with locale "en_US.utf8".
db_1    | The default database encoding has accordingly been set to "UTF8".
db_1    | The default text search configuration will be set to "english".
db_1    |
db_1    | Data page checksums are disabled.
db_1    |
db_1    | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db_1    | creating subdirectories ... ok
db_1    | selecting default max_connections ... 100
db_1    | selecting default shared_buffers ... 128MB
db_1    | selecting dynamic shared memory implementation ... posix
db_1    | creating configuration files ... ok
db_1    | running bootstrap script ... ok
db_1    | performing post-bootstrap initialization ... ok
db_1    | syncing data to disk ... ok
db_1    |
db_1    | WARNING: enabling "trust" authentication for local connections
db_1    | You can change this by editing pg_hba.conf or using the option -A, or
db_1    | --auth-local and --auth-host, the next time you run initdb.
db_1    |
db_1    | Success. You can now start the database server using:
db_1    |
db_1    |     pg_ctl -D /var/lib/postgresql/data -l logfile start
db_1    |
```

(continues on next page)

(continued from previous page)

```

db_1  | ****
db_1  | WARNING: No password has been set for the database.
db_1  | This will allow anyone with access to the
db_1  | Postgres port to access your database. In
db_1  | Docker's default configuration, this is
db_1  | effectively any other container on the same
db_1  | system.
db_1  |
db_1  | Use "-e POSTGRES_PASSWORD=password" to set
db_1  | it in "docker run".
db_1  | ****
db_1  | waiting for server to start....2018-01-18 09:51:04.629 UTC [37] LOG: ↵
db_1  | ↵listening on IPv4 address "127.0.0.1", port 5432
db_1  | 2018-01-18 09:51:04.630 UTC [37] LOG: could not bind IPv6 address "::1": ↵
db_1  | ↵Cannot assign requested address
db_1  | 2018-01-18 09:51:04.630 UTC [37] HINT: Is another postmaster already ↵
db_1  | ↵running on port 5432? If not, wait a few seconds and retry.
db_1  | 2018-01-18 09:51:04.755 UTC [37] LOG: listening on Unix socket "/var/run/
db_1  | ↵postgresql/.s.PGSQL.5432"
db_1  | 2018-01-18 09:51:04.916 UTC [38] LOG: database system was shut down at 2018-
db_1  | ↵01-18 09:51:02 UTC
db_1  | 2018-01-18 09:51:04.976 UTC [37] LOG: database system is ready to accept ↵
db_1  | ↵connections
db_1  | done
db_1  | server started
db_1  | ALTER ROLE
db_1  |
db_1  |
db_1  | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
db_1  |
db_1  | 2018-01-18 09:51:05.165 UTC [37] LOG: received fast shutdown request
db_1  | waiting for server to shut down....2018-01-18 09:51:05.224 UTC [37] LOG: ↵
db_1  | ↵aborting any active transactions
db_1  | 2018-01-18 09:51:05.226 UTC [37] LOG: worker process: logical replication ↵
db_1  | ↵launcher (PID 44) exited with exit code 1
db_1  | 2018-01-18 09:51:05.228 UTC [39] LOG: shutting down
db_1  | 2018-01-18 09:51:05.860 UTC [37] LOG: database system is shut down
db_1  | done
db_1  | server stopped
db_1  |
db_1  | PostgreSQL init process complete; ready for start up.
db_1  |
db_1  | 2018-01-18 09:51:05.947 UTC [1] LOG: listening on IPv4 address "0.0.0.0", ↵
db_1  | ↵port 5432
db_1  | 2018-01-18 09:51:05.947 UTC [1] LOG: listening on IPv6 address ":::", port ↵
db_1  | ↵5432
db_1  | 2018-01-18 09:51:06.080 UTC [1] LOG: listening on Unix socket "/var/run/
db_1  | ↵postgresql/.s.PGSQL.5432"
db_1  | 2018-01-18 09:51:06.278 UTC [55] LOG: database system was shut down at 2018-
db_1  | ↵01-18 09:51:05 UTC
db_1  | 2018-01-18 09:51:06.340 UTC [1] LOG: database system is ready to accept ↵
db_1  | ↵connections
web_1 | Performing system checks...
web_1 |
web_1 | System check identified no issues (0 silenced).
web_1 |
web_1 | You have 14 unapplied migration(s). Your project may not work properly until ↵
web_1 | ↵you apply the migrations for app(s): admin, auth, contenttypes, session (continues on next page)

```

(continued from previous page)

```
web_1 | Run 'python manage.py migrate' to apply them.
web_1 | January 18, 2018 - 10:46:37
web_1 | Django version 2.0.1, using settings 'composeexample.settings'
web_1 | Starting development server at http://0.0.0.0:8000/
web_1 | Quit the server with CONTROL-C.
```

At this point, your Django app should be running at port 8000 on your Docker host.

On Docker for Mac and Docker for Windows, go to <http://localhost:8000> on a web browser to see the Django welcome page

5.7.1.5.3 docker ps

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker>docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
30b4922c00b2	djangoapp_web	"python3 manage.py r..."	About an hour ago
↔ Up About an hour	0.0.0.0:8000->8000/tcp	djangoapp_web_1	
0883a9ef1c3b	postgres	"docker-entrypoint.s..."	2 hours ago
↔ Up 2 hours	5432/tcp	djangoapp_db_1	

5.7.1.5.4 docker-compose down

```
$ docker-compose down
```

```
Stopping djangoapp_web_1 ... done
Stopping djangoapp_db_1 ... done
Removing djangoapp_web_1    ... done
Removing djangoapp_web_run_1 ... done
Removing djangoapp_db_1      ... done
Removing network djangoapp_default
```

5.7.1.6 Compose file examples

5.7.1.6.1 Compose file examples

Compose file example 1

See also:

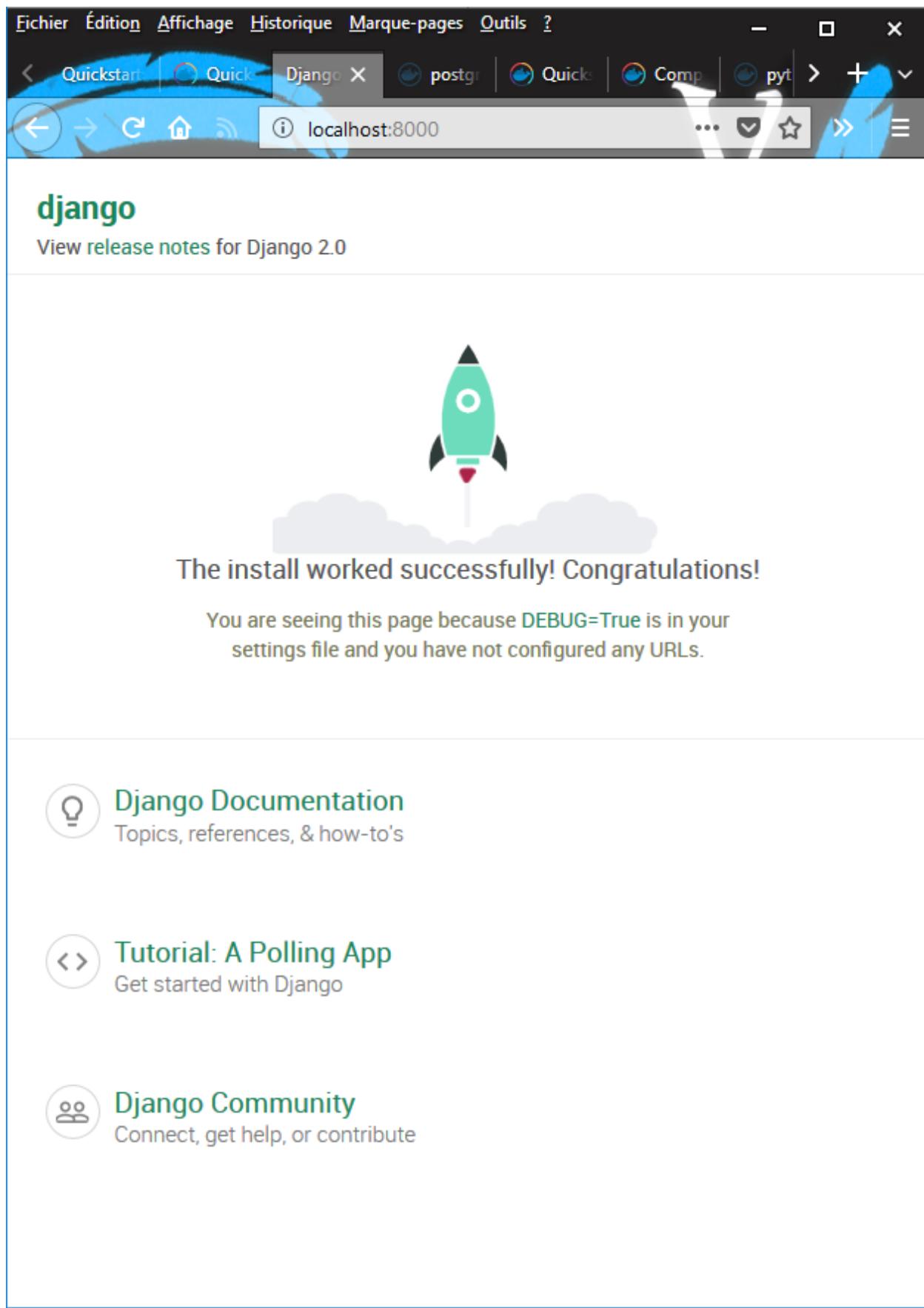
- <https://github.com/pydanny/cookiecutter-django/issues/1258>

Contents

- *Compose file example 1*

- *base.yml*

- *dev.yml*



base.yml

```
version: '3.2'
services:
  postgres:
    build: ./compose/postgres
    environment:
      - POSTGRES_USER_FILE=/run/secrets/pg_username
      - POSTGRES_PASSWORD_FILE=/run/secrets/pg_password
    secrets:
      - pg_username
      - pg_password

  django:
    command: /gunicorn.sh
    environment:
      - USE_DOCKER=$DAPI_VAR:-yes
      - DATABASE_URL=postgres://{{username}}:{{password}}@postgres:5432/{{username}}
      - SECRETS_FILE=/run/secrets/django_s
      - POSTGRES_USER_FILE=/run/secrets/pg_username
      - POSTGRES_PASSWORD_FILE=/run/secrets/pg_password
    # My Deploy
    deploy:
      replicas: 1
      restart_policy:
        condition: on-failure
      secrets:
        - pg_username
        - pg_password
        - django_s

secrets:
  django_s:
    external: True
  pg_username:
    external: True
  pg_password:
    external: True
```

dev.yml

```
version: '3.2'

volumes:
  postgres_data_dev: {}
  postgres_backup_dev: {}

services:
  postgres:
    image: apple_postgres
    volumes:
      - postgres_data_dev:/var/lib/postgresql/data
      - postgres_backup_dev:/backups

  django:
```

(continues on next page)

(continued from previous page)

```

image: apple_django
build:
  context: .
  dockerfile: ./compose/django/Dockerfile-dev
  command: /start-dev.sh
  volumes:
    - .:/app
  ports:
    - "8000:8000"
  secrets:
    - pg_username
    - pg_password
    - source: django_s
      #target: /app//.env

node:
  image: apple_node
  #user: $USER:-0
  build:
    context: .
    dockerfile: ./compose/node/Dockerfile-dev
  volumes:
    - .:/app
    - ${PWD}/gulpfile.js:/app/gulpfile.js
    # http://jdnlm.info/articles/2016/03/06/lessons-building-node-app-docker.html
    - /app/node_modules
    - /app/vendor
  command: "gulp"
  ports:
    # BrowserSync port.
    - "3000:3000"
    # BrowserSync UI port.
    - "3001:3001"

```

Compose file example 2

See also:

- <http://ramkulkarni.com/blog/docker-project-for-python3-django-and-apache2-setup/>

Contents

- *Compose file example 2*

5.7.2 gitlab ARM

See also:

- <https://gitlab.com/ulm0/gitlab#install-gitlab-using-docker-compose>

Contents

- *gitlab ARM*
 - *Overview of Docker Compose*

5.7.2.1 Overview of Docker Compose

CHAPTER 6

compose-file

See also:

- <https://docs.docker.com/compose/overview>
- <https://docs.docker.com/compose/reference>
- <https://docs.docker.com/compose/compose-file>
- <https://docs.docker.com/compose/samples-for-compose>
- <https://github.com/search?q=in%3Apath+docker-compose.yml+extension%3Ayml&type=Code>
- [*docker-compose versions*](#)

Contents

- [*compose-file*](#)
 - [*Versions*](#)
 - * [*3.7*](#)
 - * [*3.6*](#)

6.1 Versions

See also:

<https://docs.docker.com/compose/compose-file/compose-versioning>

6.1.1 3.7

See also:

- <https://docs.docker.com/compose/compose-file/compose-versioning/#version-37>

An upgrade of version 3 that introduces new parameters.

It is only available with Docker Engine version *18.06.0* and higher.

Introduces the following additional parameters:

- init in service definitions
- rollback_config in deploy configurations
- Support for extension fields at the root of service, network, volume, secret and config definitions

6.1.2 3.6

See also:

- <https://docs.docker.com/compose/compose-file/compose-versioning/#version-36>

An upgrade of version 3 that introduces new parameters. It is only available with Docker Engine version 18.02.0 and higher.

Introduces the following additional parameters:

```
tmpfs size for tmpfs-type mounts
```

Bonnes pratiques Docker

Contents

- *Bonnes pratiques Docker*
 - *actu.alfa-safety.fr*
 - *Best practices for writing Dockerfiles*
 - *Best practices for writing Dockerfiles from Nick Janetakis*

7.1 actu.alfa-safety.fr

See also:

- <https://actu.alfa-safety.fr/devops/docker-en-production/>

Docker est largement utilisé en développement mais bien souvent les choses se compliquent en production:

- d'abord l'application ne fonctionne pas toujours correctement en prod,
- les performances ne suivent pas,
- ensuite on s'aperçoit que l'on a oublié de prendre en compte un certain nombre d'éléments indispensables en production: monitoring, scalabilité, contraintes réseaux.

La facilité est alors de dire : Docker fonctionne bien en Dev mais n'est pas un outil adapté à la production. **Bien au contraire**, Docker en production doit permettre de faciliter et sécuriser les déploiements tout en rendant votre application scalable.

Mais pour cela, il faut bien fonctionner en mode Devops et respecter un certain nombre de bonnes pratiques. C'est en tant que telle une compétence ou expertise Docker en production qu'il faut développer.

Enfin quand votre production atteint une certaine complexité, et que le nombre de conteneurs que vous gérez se compte en dizaines, il faut envisager de passer sur un orchestrateur de conteneurs.

Avant d'attaquer le vif du sujet, vous pouvez revenir sur notre précédent article sur les bases de Docker.

7.2 Best practices for writing Dockerfiles

See also:

- https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/
- <https://docs.docker.com/engine/reference/builder/>

Docker can build images automatically by reading the instructions from a Dockerfile, a text file that contains all the commands, in order, needed to build a given image. Dockerfiles adhere to a specific format and use a specific set of instructions.

You can learn the basics on the [Dockerfile Reference page](#). If you're new to writing Dockerfiles, you should start there.

This document covers the best practices and methods recommended by Docker, Inc. and the Docker community for building efficient images.

To see many of these practices and recommendations in action, check out the Dockerfile for `buildpack-deps`.

Note: for more detailed explanations of any of the Dockerfile commands mentioned here, visit the [Dockerfile Reference page](#).

7.2.1 9-pillars-of-containers-best-practices

See also:

- <https://containerjournal.com/2018/10/16/9-pillars-of-containers-best-practices/>

7.3 Best practices for writing Dockerfiles from Nick Janetakis

7.3.1 Docker Best practices from Nick Janetakis

See also:

- [Nick Janetakis](#)
- <https://nickjanetakis.com/blog/best-practices-when-it-comes-to-writing-docker-related-files>
- <https://github.com/nickjj/docker-web-framework-examples>
- <https://dev.to/nickjj/best-practices-when-it-comes-to-writing-docker-related-files-ek3>
- <https://github.com/nickjj/docker-web-framework-examples>

Contents

- [Docker Best practices from Nick Janetakis](#)
 - [Dockerfile](#)
 - [docker-compose.yml](#)

- `.dockerignore`
- `Example Apps for Popular Web Frameworks`
- `Flask example`
 - * `Flask Dockerfile`
 - * `.env file`
 - * `Flask docker-compose.yml`
 - * `hello/app.py`
 - * `.gitignore`

7.3.1.1 Dockerfile

- Use Alpine as a base image unless you can't due to technical reasons
- Pin versions to at least the minor version, example: 2.5-alpine not 2-alpine
- Add a maintainer LABEL to keep tabs on who initially made the image
- Only include ARG and ENV instructions if you really need them
- Use /app to store your app's code and set it as the WORKDIR (if it makes sense)
- When installing packages, take advantage of Docker's layer caching techniques
- If your app is a web service, *EXPOSE 8000 unless you have a strong reason not to*
- Include a wget driven HEALTHCHECK (if it makes sense)
- Stick to the [] syntax when supplying your CMD instructions

7.3.1.2 docker-compose.yml

- List your services in the order you expect them to start
- Alphabetize each service's properties
- Double quote all strings and use {} for empty hashes / dictionaries
- Pin versions to at least the minor version, example: 10.4-alpine not 10-alpine
- Use . instead of \$PWD for when you need the current directory's path
- Prefer build: “.” unless you need to use args or some other sub-property
- If your service is a web service, publish port 8000 unless it doesn't make sense to

7.3.1.3 .dockerignore

- Don't forget to create this file :D
- Don't forget to add the .git folder
- Don't forget to add any sensitive files such as .env.production

7.3.1.4 Example Apps for Popular Web Frameworks

I've put together a few example applications that stick to these best practices.

You can find them all on <https://github.com/nickjj/docker-web-framework-examples> Fully working Docker Compose based examples that you can reference:

- Flask
- Node / Express
- Phoenix
- Rails
- Webpack

If you don't see your favorite web framework listed above, open up a PR! This repo is meant to be a community effort where we can work together to make high quality example apps that demonstrate Dockerizing popular web frameworks and libraries.

7.3.1.5 Flask example

See also:

- <https://github.com/nickjj/docker-web-framework-examples/tree/master/flask>

7.3.1.5.1 Flask Dockerfile

```
FROM python:2.7-alpine
LABEL maintainer="Nick Janetakis <nick.janetakis@gmail.com>

# If you plan to use PostgreSQL then you must add this package: postgresql-dev.
RUN apk update && apk add build-base

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000
HEALTHCHECK CMD wget -q -O /dev/null http://localhost:8000/healthy || exit 1

CMD ["gunicorn", "-c", "python:config.gunicorn", "hello.app:create_app()"]
```

7.3.1.5.2 .env file

```
COMPOSE_PROJECT_NAME=flaskhello
PYTHONUNBUFFERED=true
```

7.3.1.5.3 Flask docker-compose.yml

```
version: "3.6"

services:
  web:
    build: "."
    command: >
      gunicorn --reload -c "python:config.gunicorn" "hello.app:create_app()"
    env_file:
      - ".env"
    ports:
      - "8000:8000"
    volumes:
      - ".:/app"
```

7.3.1.5.4 hello/app.py

```
from flask import Flask
from werkzeug.debug import DebuggedApplication

def create_app(settings_override=None):
    """
    Create a Flask application using the app factory pattern.

    :param settings_override: Override settings
    :return: Flask app
    """
    app = Flask(__name__, instance_relative_config=True)

    app.config.from_object('config.settings')
    app.config.from_pyfile('settings.py', silent=True)

    if settings_override:
        app.config.update(settings_override)

    if app.debug:
        app.wsgi_app = DebuggedApplication(app.wsgi_app, evalex=True)

    @app.route('/')
    def index():
        return 'Hello world with DEBUG={0}'.format(app.config['DEBUG'])

    @app.route('/healthy')
    def healthy():
        return ''

    return app
```

7.3.1.5.5 .gitignore

See also:

- <https://www.gitignore.io/api>

```
# Created by https://www.gitignore.io/api/python,osx

### OSX ###
*.DS_Store
.AppleDouble
.LSOVERRIDE

# Icon must end with two \r
Icon

# Thumbnails
._*

# Files that might appear in the root of a volume
.DocumentRevisions-V100
.fsevents.d
.Spotlight-V100
.TemporaryItems
.Trashes
.VolumeIcon.icns
.com.apple.timemachine.donotpresent

# Directories potentially created on remote AFP share
.AppleDB
.AppleDesktop
Network Trash Folder
Temporary Items
.apdisk

### Python ###
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
*.egg-info/
.installed.cfg
*.egg
```

(continues on next page)

(continued from previous page)

```
# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.coverage
.coverage./*
.cache
.pytest_cache/
nosetests.xml
coverage.xml
*.cover
.hypothesis/

# Translations
*.mo
*.pot

# Flask stuff:
instance/settings.py
.webassets-cache

# Scrapy stuff:
.scrapy

# celery beat schedule file
celerybeat-schedule./*

# End of https://www.gitignore.io/api/python,osx
```


CHAPTER 8

Docker machine

See also:

- <https://docs.docker.com/machine/overview/>

Contents

- *Docker machine*

CHAPTER 9

Docker swarm

See also:

- <https://docs.docker.com/engine/swarm/>

9.1 Docker swarm articles

See also:

- <https://docs.docker.com/engine/swarm/>

9.1.1 Docker swarm articles 2018

See also:

- <https://docs.docker.com/engine/swarm/>

9.1.1.1 Only one host for production environment. What to use: docker-compose or single node swarm ?

See also:

- *Bret Fischer*
- <https://github.com/BretFisher/ama/issues/8>
- <https://twitter.com/BretFisher>

Contents

- *Only one host for production environment. What to use: docker-compose or single node swarm ?*

- *Question*
- *Response*

9.1.1.1 Question

We have recently moved all our corporative services to run ONE DigitalOcean server having all services in a docker environment: redmine, dokuwiki, opends, mattermost, a docker registry, portainer,...

The way we did it, was creating all the needed docker-compose files (one by service, and having all the needed containers in each one: RoR+postgresql, Node+Mongo+redis,...), add all the needed mountpoints for the volumes (and almost all containers must be persistent), and include the option in all of them with “restart: always”.

All this apps were started with ‘docker-compose -d up’, and in this way this only ONE server is able to run all services (and all of them get started with server startup). We don’t need a cluster right now.

We don’t know if this approach is a good one or it shouldn’t be used for production (and why in this case). We want to have one server to pay the less as possible, and taking into account that it can manage all our apps. Should we create a swarm, move all containers to be swarm services, but only have one manager and no workers? I would be that approach a better option?

If this is true, what should we use to replace the use of jwilder/nginx-proxy (and docker-letsencrypt-nginx-proxy-companion) to manage http redirections and automatic generation of letsencrypt certificates.

Thanks in advance!

9.1.1.2 Response

I always recommend single-node Swarm with the assumptions you know the risks of a single node of anything, and you’re backing up persistent data, keys/secrets, etc.

My top reasons for a single-node Swarm over docker-compose:

- It only takes a single command to create a Swarm from that docker host docker swarm init.
- It saves you from needing to manually install/update docker-compose on that server. Docker engine is installable and updatable via common Linux package managers (apt, yum) via <https://store.docker.com> but docker-compose is not.
- When you’re ready to become highly-available, you won’t need to start from scratch. Just add two more nodes to a well-connected network with the 1st node. Ensure firewall ports are open between them. Then use docker swarm join-token manager on 1st node and run that output on 2nd/3rd. Now you have a fully redundant raft log and managers. Then you can change your compose file for multiple replicas of each of your services and re-apply with docker stack deploy again and you’re playin’ with the big dogs!
- You get a lot of extra features out-of-the-box with Swarm, including secrets, configs, auto-recovery of services, rollbacks, healthchecks, and ability to use Docker Cloud Swarms BYOS to easily connect to swarm without SSH.
- Healthchecks, healthchecks, healthchecks. docker run and docker-compose won’t re-create containers that failed a built-in healthcheck. You only get that with Swarm, and it should always be used for production on all containers.
- Rolling updates. Swarm’s docker service update command (which is also used by docker stack deploy when updating yaml changes) has TONS of options for controlling how you replace containers during an update. If you’re running your own code on a Swarm, updates will be often, so you want to make sure the process is smooth, depends on healthchecks for being “ready”, maybe starts a new one first before turning off old container, and rolls back if there’s a problem. None of that happens without Swarm’s orchestration and scheduling.

- Local docker-compose for development works great in the workflow of getting those yaml files into production Swarm servers.
- Docker and Swarm are the same daemon, so no need to worry about version compatibility of production tools. Swarm isn't going to suddenly make your single production server more complex to manage and maintain.

There's more, but that's my big ticket heavy hitters!

CHAPTER 10

Docker commands

See also:

- <https://docs.docker.com/engine/reference/commandline/docker/#description>

Contents

- *Docker commands*
 - *docker help*
 - *docker attach*
 - *docker build*
 - * *Description*
 - * *docker build –no-cache*
 - *docker commit*
 - *docker cp*
 - *docker diff*
 - *docker exec*
 - *docker export*
 - *docker history*
 - *docker inspect*
 - *docker images*
 - *docker kill*
 - *docker login*
 - *docker logs*

```
* Description
- docker ps
  * docker ps -filter
- docker pull
- docker rename
- docker run
  * detach-keys
  * downloading images
  * labels
  * entrypoint
- docker search
  * Description
- docker system
  * docker system prune
- docker stop
- docker tag
- docker volume
```

10.1 docker help

- <https://docs.docker.com/engine/reference/commandline/help/>

```
docker help
```

```
Usage: docker COMMAND

A self-sufficient runtime for containers

Options:
  --config string          Location of client config files (default "/home/pvergайн/.
                           ↪docker")
  -D, --debug              Enable debug mode
  -H, --host list           Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal"
                           ↪") (default "info")
  --tls                   Use TLS; implied by --tlsverify
  --tlscacert string       Trust certs signed only by this CA (default "/home/
                           ↪pvergайн/.docker/ca.pem")
  --tlscert string         Path to TLS certificate file (default "/home/pvergайн/.
                           ↪docker/cert.pem")
  --tlskey string          Path to TLS key file (default "/home/pvergайн/.docker/key.
                           ↪pem")
  --tlsverify              Use TLS and verify the remote
  -v, --version             Print version information and quit
```

(continues on next page)

(continued from previous page)

Management Commands:	
config	Manage Docker configs
container	Manage containers
image	Manage images
network	Manage networks
node	Manage Swarm nodes
plugin	Manage plugins
secret	Manage Docker secrets
service	Manage services
swarm	Manage Swarm
system	Manage Docker
trust	Manage trust on Docker images
volume	Manage volumes
Commands:	
attach	Attach local standard <code>input</code> , <code>output</code> , and error streams to a running container
build	Build an image from a Dockerfile
commit	Create a new image from a container's changes
cp	Copy files/folders between a container and the local filesystem
create	Create a new container
diff	Inspect changes to files or directories on a container's filesystem
events	Get real time events from the server
exec	Run a command in a running container
export	Export a container's filesystem as a tar archive
history	Show the history of an image
images	List images
import	Import the contents from a tarball to create a filesystem image
info	Display system-wide information
inspect	Return low-level information on Docker objects
kill	Kill one or more running containers
load	Load an image from a tar archive or STDIN
login	Log in to a Docker registry
logout	Log out from a Docker registry
logs	Fetch the logs of a container
pause	Pause all processes within one or more containers
port	List port mappings or a specific mapping for the container
ps	List containers
pull	Pull an image or a repository from a registry
push	Push an image or a repository to a registry
rename	Rename a container
restart	Restart one or more containers
rm	Remove one or more containers
rmi	Remove one or more images
run	Run a command in a new container
save	Save one or more images to a tar archive (streamed to STDOUT by default)
search	Search the Docker Hub for images
start	Start one or more stopped containers
stats	Display a live stream of container(s) resource usage statistics
stop	Stop one or more running containers
tag	Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top	Display the running processes of a container
unpause	Unpause all processes within one or more containers
update	Update configuration of one or more containers
version	Show the Docker version information
wait	Block until one or more containers stop, then print their exit codes

(continues on next page)

(continued from previous page)

```
Run 'docker COMMAND --help' for more information on a command.
```

10.2 docker attach

See also:

- <https://docs.docker.com/engine/reference/commandline/attach/>
- *View only the IDs of the containers*

10.3 docker build

See also:

- <https://docs.docker.com/engine/reference/commandline/build/>
- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#build-context

10.3.1 Description

The docker build command builds Docker images from a **Dockerfile** and a **context**. A build's context is the set of files located in the specified PATH or URL. The build process can refer to any of the files in the context. For example, your build can use a COPY instruction to reference a file in the context. The URL parameter can refer to three kinds of resources:

- Git repositories,
- pre-packaged tarball contexts
- and plain text files.

```
docker build --tag gdevops/mydjango4 .
```

10.3.2 docker build –no-cache

You can force a rebuild with docker build –no-cache

10.4 docker commit

See also:

- <https://docs.docker.com/engine/reference/commandline/commit/>
- *Commit our changes into a new image*
- *Exploring a crashed container (docker commit + docker run -ti –entrypoint)*

10.5 docker cp

See also:

- <https://docs.docker.com/engine/reference/commandline/cp/>
- *Commit our changes into a new image*

10.6 docker diff

See also:

- <https://docs.docker.com/engine/reference/commandline/diff/>
- *Inspect the changes*
- *Viewing filesystem changes*

10.7 docker exec

See also:

- <https://docs.docker.com/engine/reference/commandline/exec/>
- *How to run a shell in our running container ?*

Examples:

```
docker run -d -p 8000:5000 -p 8001:5001 --name myany test_sqlanywhere:latest
docker exec -ti myany bash
```

10.8 docker export

See also:

- <https://docs.docker.com/engine/reference/commandline/export/>
- *Obtaining a complete dump (docker export)*

10.9 docker history

See also:

- <https://docs.docker.com/engine/reference/commandline/history/>
- *Using image and viewing history*

```
docker history gdevops/mydjango4
```

10.10 docker inspect

See also:

- <https://docs.docker.com/engine/reference/commandline/inspect/>
- *Naming and inspecting containers*
- *Inspecting a container*
- *Using --format*

10.11 docker images

See also:

- <https://docs.docker.com/engine/reference/commandline/images/>
- *Showing current images*

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
gdevops/mydjango4	latest	105b33d63fe8	5 minutes ago	984MB
python	3.6	29d2f3226daf	3 weeks ago	911MB

10.12 docker kill

See also:

- <https://docs.docker.com/engine/reference/commandline/kill/>
- *Stop our container*

10.13 docker login

See also:

- <https://docs.docker.com/engine/reference/commandline/login/>
- *Logging into our Docker Hub account*

10.14 docker logs

See also:

- <https://docs.docker.com/engine/reference/commandline/logs/>
- *View the logs of a container*

10.14.1 Description

Fetch the logs of a container.

Usage:

```
docker logs [OPTIONS] CONTAINER
```

```
docker logs apache
```

10.15 docker ps

See also:

- <https://docs.docker.com/engine/reference/commandline/ps/>
- *List running containers*
- *View only the IDs of the containers*
- *Using labels to select containers (docker ps --filter)*

10.15.1 docker ps –filter

See also:

- *Using labels to select containers (docker ps --filter)*

10.16 docker pull

See also:

- <https://docs.docker.com/engine/reference/commandline/pull/>
- *Downloading images*

10.17 docker rename

See also:

- <https://docs.docker.com/engine/reference/commandline/rename/>
- *Renaming containers*

10.18 docker run

See also:

- <https://docs.docker.com/engine/reference/commandline/run/>
- *Avril 2018 container training from Jérôme Petazzoni*

- *Specifying a name*
- *Using labels*
- *Detaching from a container*
- *Exploring a crashed container (docker commit + docker run -ti --entrypoint)*

10.18.1 detach-keys

See also:

- *Detaching from a container*

10.18.2 downloading images

See also:

- *Downloading images*

10.18.3 labels

See also:

- *Using labels*

10.18.4 entrypoint

See also:

- *Exploring a crashed container (docker commit + docker run -ti --entrypoint)*

10.19 docker search

See also:

- <https://docs.docker.com/engine/reference/commandline/search/>

10.19.1 Description

Search the Docker Hub for images.

Usage:

```
docker search [OPTIONS] TERM
```

```
docker search apache
```

NAME	STARS	OFFICIAL	DESCRIPTION
tomcat	2063	[OK]	AUTOMATED Apache Tomcat is an open source link
httpd	2038	[OK]	The Apache HTTP Server Project link
cassandra	868	[OK]	Apache Cassandra is an open-source link
maven	698	[OK]	Apache Maven is a software project link
solr	586	[OK]	Solr is the popular, blazing-fast, link
open sour...	586	[OK]	
zookeeper	484	[OK]	Apache ZooKeeper is an open-source link
eboraas/apache-php	139		PHP5 on Apache (with SSL support), link
eboraas/apache	90		Apache (with SSL support), built on link
eboraas/php-apache-dev	78		[OK]
webdevops/php-apache	74		PHP with Apache for Development (eg. link)
webdevops/php	74		[OK]
groovy	58	[OK]	Apache Groovy is a multi-faceted link
language fo...	58		
tomee		[OK]	Apache TomEE is an all-Apache Java EE link
certif...	56		This is docker images of Ubuntu 14.04 link
nimmis/apache-php5	53		[OK]
apacheignite/ignite	44		Apache Ignite In-Memory docker image. link
bitnami/apache	42		[OK]
linuxserver/apache	18		Bitnami Apache Docker Image link
apache/nutch	15		[OK]
land1internet/ubuntu-16-apache-php-7.0	13		An Apache container, brought to you by link
webdevops/apache	11		Apache Nutch link
antage/apache2-php5	10		[OK]
newdeveloper/apache-php	3		ubuntu-16-apache-php-7.0 link
lephare/apache	4		Apache container link
mastertinner/apache-directory-index-resource	1		[OK]
secoresearch/apache-varnish	0		Apache container link
jelastic/apachephp	0		apache-php7.2 link
serve...	0		A Concourse resource for the apache link

10.20 docker system

See also:

- <https://docs.docker.com/engine/reference/commandline/system/>

10.20.1 docker system prune

```
docker system prune
```

10.21 docker stop

See also:

- <https://docs.docker.com/engine/reference/commandline/stop/>
- *Stop our container*

10.22 docker tag

See also:

- <https://docs.docker.com/engine/reference/commandline/tag/>
- *Tagging images*

10.23 docker volume

See also:

- <https://docs.docker.com/engine/reference/commandline/volume/>

```
$ docker help volume
```

```
Usage: docker volume COMMAND
```

```
Manage volumes
```

```
Options:
```

```
Commands:
```

create	Create a volume
inspect	Display detailed information on one or more volumes
ls	List volumes
prune	Remove all unused local volumes
rm	Remove one or more volumes

```
Run 'docker volume COMMAND --help' for more information on a command.
```

CHAPTER 11

Dockerfile

See also:

- <https://docs.docker.com/engine/reference/builder/>
- <https://docs.docker.com/engine/deprecated/>

Contents

- *Dockerfile*
 - *Deprecated*
 - * *MAINTAINER*

11.1 Deprecated

11.1.1 MAINTAINER

- <https://docs.docker.com/engine/deprecated/#maintainer-in-dockerfile>

MAINTAINER was an early very limited form of LABEL which should be used instead.

The recommended solution is to use LABEL instead, e.g. LABEL authors="first author,second author"

CHAPTER 12

Docker network

See also:

- <https://github.com/vrde/notes/tree/master/docker-playground>
- <https://github.com/docker/labs/tree/master/networking>

Contents

- *Docker network*
 - *Las networking*

12.1 Las networking

See also:

- <https://github.com/docker/labs/tree/master/networking>

CHAPTER 13

Volumes Docker

See also:

- <https://docs.docker.com/engine/admin/volumes/volumes/>
- <http://www.lemagit.fr/conseil/Docker-quelles-sont-les-options-pour-le-stockage-persistant>
- <http://xataz.developpez.com/tutoriels/utilisation-docker/>

Contents

- *Volumes Docker*
 - *Use volumes*
 - *Create and manage volumes*
 - * *docker volume create*
 - * *docker volume ls*

13.1 Use volumes

Estimated reading time: 12 minutes

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.

While bind mounts are dependent on the directory structure of the host machine, volumes are completely managed by Docker.

Volumes have several advantages over bind mounts:

- Volumes are easier to back up or migrate than bind mounts.
- You can manage volumes using Docker CLI commands or the Docker API.
- Volumes work on both Linux and Windows containers.

- Volumes can be more safely shared among multiple containers.
- Volume drivers allow you to store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
- A new volume's contents can be pre-populated by a container.

In addition, volumes are often a better choice than persisting data in a container's writable layer, because using a volume does not increase the size of containers using it, and the volume's contents exist outside the lifecycle of a given container.

13.2 Create and manage volumes

Unlike a bind mount, you can create and manage volumes outside the scope of any container.

13.2.1 docker volume create

Create a volume:

```
docker volume create my-vol
```

13.2.2 docker volume ls

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\pipenv>
→ docker volume ls
```

DRIVER	VOLUME NAME
local	03f1a1ed0555015e51863dbed5f6c7847099fd33449b9d83919bd7028cdfdd9b
local	4bc5fb631c6af81f5ba84a8465b3c2805ca713541fe736faf3a232ef4b24ae72
local	56295a3bb8a90d260864c258e6b174755338543a614f206e5082c066d22eb197
local	67871ba2f3b3a9e75fdbfcf2fe5ec36ba7a10cd5930a60e8227abc7110e62ca4
local	b6432532ff915143ede0b7169abf7690790ce0227277013d7c5ab00007d68703
local	bbef076d429a90ca7bfd7a751d7c8aa1ea3d08e0b7b4036bb296681545940a0b
local	bf69b1f1164c09d7dc0f3e6011f3116e7bc197e0e9341e645a15fdc7566489f3
local	cee0d9fed75150bda5e6b32c5eeaad4e433afe01165bf822eae8413b1f4e861
local	pgdata
local	postgresql_postgres_data
local	vote_db-data

CHAPTER 14

Registry

See also:

- <https://docs.docker.com/registry/introduction/>

Contents

- *Registry*
 - *Definition*
 - *Understanding image naming*
 - *Use cases*
 - *Implementations*
 - *Examples*

14.1 Definition

The registry is the “ship” part of the build, ship, run workflow.

You package your app in a Docker image using a Dockerfile and docker image build, and the output is an image on your machine (or the CI server that ran the build).

To make the image available to other users, you ship it to a registry with docker image push. **The default registry is Docker Hub, which is a free public registry service.**

If you want to keep your images private, so they’re only accessible within your own network, you can use a commercial registry like Docker Trusted Registry - which also provides security scanning and image signing.

14.2 Understanding image naming

See also:

- <https://docs.docker.com/registry/introduction/>

Image names as used in typical docker commands reflect their origin:

- **docker pull** ubuntu instructs docker to pull an image named ubuntu from the official Docker Hub. This is simply a shortcut for the longer docker pull docker.io/library/ubuntu command
- docker pull myregistrydomain:port/foo/bar instructs docker to contact the registry located at myregistrydomain:port to find the image foo/bar

14.3 Use cases

Running your own Registry is a great solution to integrate with and complement your CI/CD system. In a typical workflow, a commit to your source revision control system would trigger a build on your CI system, which would then push a new image to your Registry if the build is successful. A notification from the Registry would then trigger a deployment on a staging environment, or notify other systems that a new image is available.

It's also an essential component if you want to quickly deploy a new image over a large cluster of machines.

Finally, it's the best way to distribute images inside an isolated network. Requirements

You absolutely need to be familiar with Docker, specifically with regard to pushing and pulling images. You must understand the difference between the daemon and the cli, and at least grasp basic concepts about networking.

Also, while just starting a registry is fairly easy, operating it in a production environment requires operational skills, just like any other service. You are expected to be familiar with systems availability and scalability, logging and log processing, systems monitoring, and security 101.

Strong understanding of http and overall network communications, plus familiarity with golang are certainly useful as well for advanced operations or hacking.

14.4 Implementations

14.4.1 Docker Registry implementations

14.4.1.1 Gitlab Container Registry

See also:

- https://docs.gitlab.com/ee/user/project/container_registry.html
- https://docs.gitlab.com/ee/ci/docker/using_docker_build.html#using-the-gitlab-container-registry

Contents

- *Gitlab Container Registry*
 - *Historique*
 - * *2016-05-23 : GitLab Container Registry*

- * *Introduction*
- * *Docker Basics*
- * *Summary*
- *Administration*
- *Examples*

14.4.1.1.1 Historique

2016-05-23 : GitLab Container Registry

See also:

- <https://about.gitlab.com/2016/05/23/gitlab-container-registry/>

Introduction

Yesterday we released GitLab 8.8, super powering GitLab's built-in continuous integration. With it, you can build a pipeline in GitLab, visualizing your builds, tests, deploys and any other stage of the life cycle of your software. Today (and already in GitLab 8.8), we're releasing the next step: GitLab Container Registry.

GitLab Container Registry is a secure and private registry for Docker images. Built on open source software, GitLab Container Registry isn't just a standalone registry; it's completely integrated with GitLab.

GitLab is all about having a single, integrated experience and our registry is no exception. You can now easily use your images for GitLab CI, create images specific for tags or branches and much more.

Our container registry is the first Docker registry that is fully integrated with Git repository management and comes out of the box with GitLab 8.8. So if you've upgraded, you already have it! This means our integrated Container Registry requires no additional installation. It allows for easy upload and download of images from GitLab CI. And it's free.

Docker Basics

The main component of a Docker-based workflow is an **image**, which contains everything needed to run an application. Images are often created automatically as part of continuous integration, so they are updated whenever code changes. When images are built to be shared between developers and machines, they need to be stored somewhere, and that's where a container registry comes in.

The registry is the place to store and tag images for later use. Developers may want to maintain their own registry for private, company images, or for throw-away images used only in testing.

Using GitLab Container Registry means you don't need to set up and administer yet another service, or use a public registry.

Summary

GitLab Container Registry is the latest addition to GitLab's integrated set of tools for the software development life cycle and comes with GitLab 8.8 and up.

With GitLab Container Registry, testing and deploying Docker containers has never been easier. GitLab Container Registry is available on-premises in GitLab CE and GitLab EE at no additional cost and installs in the same infrastructure as the rest of your GitLab instance.

Container Registry is enabled on GitLab.com; it's completely free, and you can start using it right now!

14.4.1.1.2 Administration

See also:

- https://docs.gitlab.com/ce/administration/container_registry.html

14.4.1.1.3 Examples

Gitlab Container Registry examples

Un registry Docker privé avec GitLab

See also:

- <https://lumao.eu/post/gitlab-private-registry-docker/>

14.5 Examples

14.5.1 Docker Registry examples

14.5.1.1 Running Your Own Registry

See also:

- <https://blog.sixeyed.com/windows-weekly-dockerfile-20-running-your-own-registry/>

CHAPTER 15

Glossaire Docker

See also:

- <https://docs.anaconda.com/anaconda/glossary>

Agile Software Development A set of concepts, practices and principles for the development of software under which both requirements and the software that meets them evolve during the development life-cycle by processes of collaboration, as opposed to being defined at milestones within it

Containers Running instances of Docker images — containers run the actual applications. A container includes an application and all of its dependencies. It shares the kernel with other containers, and runs as an isolated process in user space on the host OS. You created a container using docker run which you did using the alpine image that you downloaded. A list of running containers can be seen using the docker ps command.

Docker

Définition 1 (anglais) Docker allows us to easily create clean, pre-installed images of our application in an isolated state, like a binary application build, rather than having to worry about virtual environments and system packages of whatever server we are deploying to. This build can then be tested and deployed as if it was an isolated artifact in and of itself.

Source: <https://peakwinter.net/blog/modern-devops-django/>

Définition 2 With Docker, you can run your Django project on an Ubuntu server in a container on your laptop, and because Docker is available for Mac, Linux, and Windows, your choice of operating system really comes down to preference. When it comes time to push your code to a staging or production server, you can be sure it'll run exactly the same as it did on your laptop, because you can configure a Dockerfile to exactly match these environments.

Source: <https://medium.com/@adamzeitcode/a-simple-recipe-for-django-development-in-docker-bonus-testing-with-selen>

Docker daemon The background service running on the host that manages building, running and distributing Docker containers.

Docker client The command line tool that allows the user to interact with the Docker daemon.

docker-compose.yml

Definition 1 (français) Le docker compose est un fichier de configuration de l'ensemble des Dockers que vous souhaitez déployer pour votre application, il sert à les déployer et à gérer les liens entre les conteneurs ainsi que les volumes de data.

Definition 2 (anglais) The file where you can set up your database, automatically start your server when you start your container, and cool stuff like that.

Source: <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>

Définition 3 (anglais) Docker Compose lets you run more than one container in a Docker application. It's especially useful if you want to have a database, like Postgres, running in a container alongside your web app. (Docker's overview of Compose is helpful.) Compose allows you to define several services that will make up your app and run them all together.

Source: <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>

Dockerfile

Definition 1 (français) C'est le fichier texte qui décrit la configuration de votre docker, en général , on part d'une image standard et on ajoute les éléments propres à la configuration de l'application que l'on veut déployer ; une fois le Dockerfile finalisé, on *build* le conteneur.

Definition 2 (anglais) The name of the file that contains the instructions for setting up your image. Source: <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>

Docker image

Definition 1 (français) C'est l'élément de base d'un docker, on utilise une Docker image à deux stades :

- Au départ, on va chercher une image de base standard pour l'applicatif choisi (Nginx, Php, Redis), le plus souvent dans un repository public, on complète ensuite cette image standard des éléments de configuration de votre application, vous pouvez ensuite enregistrer la nouvelle image dans un repository public, ou privé,

Definition 2 (anglais) The file system and configuration of our application which are used to create containers. To find out more about a Docker image, run:

```
docker inspect alpine
```

In the demo above, you used the docker pull command to download the alpine image. When you executed the command docker run hello-world, it also did a docker pull behind the scenes to download the hello-world image.

Definition 3 (anglais) A **lightweight, stand-alone, executable package that includes everything needed to run a piece of software**. You will set up a specific image for each project you work on that will tell Docker which packages your project needs, where your code lives, etc.

Source: <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>

Docker Store A registry of Docker images, where you can find trusted and enterprise ready containers, plugins, and Docker editions. You'll be using this later in this tutorial.

hyperviseur

Hyperviseur En informatique, un hyperviseur est une plate-forme de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps.

Hyper-V Microsoft Hyper-V, codenamed Viridian and formerly known as Windows Server Virtualization, is a native hypervisor; it can create virtual machines on x86-64 systems running Windows.

Hyper-V, également connu sous le nom de Windows Server Virtualisation, est un système de virtualisation basé sur un hyperviseur 64 bits de la version de Windows Server 2008.

Orchestrator de conteneurs L'orchestrator est un peu au conteneur ce que vSphere/vCenter est à VMware pour des VMs, c'est le logiciel de gestion de l'ensemble des conteneurs sur un pool de ressources serveurs, avec davantage de fonctionnalités que vSphere/vCenter. C'est en quelque sorte un PaaS pour les conteneurs

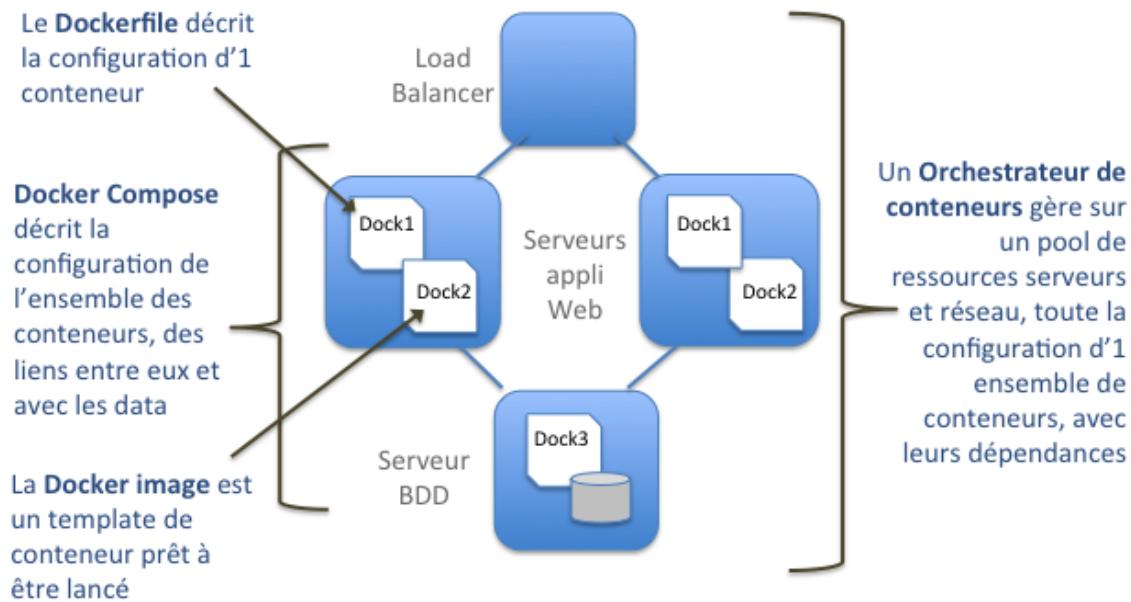


Fig. 1: Un orchestrator gère un pool de ressources serveurs et réseau .. seealso:: <https://actu.alfa-safety.fr/devops/docker-en-production/>

reverse proxy

proxy inverse Un proxy inverse (reverse proxy) est un type de serveur, habituellement placé en frontal de serveurs web. Contrairement au serveur proxy qui permet à un utilisateur d'accéder au réseau Internet, le proxy inverse permet à un utilisateur d'Internet d'accéder à des serveurs internes, une des applications courantes du proxy inverse est la répartition de charge (load-balancing).

Le proxy inverse est installé du côté des serveurs Internet. L'utilisateur du Web passe par son intermédiaire pour accéder aux applications de serveurs internes. Le proxy inverse est parfois appelé substitut (surrogate)

See also:

https://fr.wikipedia.org/wiki/Proxy_inverse

Essaim

Swarm

swarm A swarm is a cluster of one or more Docker Engines running in swarm mode. En français *swarm* est un essaim

Fig. 2: Un essaim de docker engines

Virtual machine Have you ever seen someone boot up Windows on a Mac ? That process of running one complete OS on top of another OS called running a **virtual machine**.

See also:

<http://erick.matsen.org/2018/04/19/docker.html>

CHAPTER 16

docker FAQ

See also:

- <https://docs.docker.com/compose/faq/>

Contents

- *docker FAQ*
 - *How to delete all your docker images ?*
 - *How to run a shell in our running container ?*
 - *How to delete stopped containers ?*
 - *Where can I find example compose files ?*

16.1 How to delete all your docker images ?

```
docker rm $(docker ps -a -q)
```

16.2 How to run a shell in our running container ?

See also:

- *Getting inside a container*
- *Getting a shell in a running container*
- *Getting a shell in a stopped container*

There are 2 methods:

- *docker exec*

```
$ docker exec -ti ticktock sh
```

- overriding the Dockerfile entrypoint see <https://avril2018.container.training/intro.yml.html#194>

```
$ docker run -it --entrypoint bash figlet
```

16.3 How to delete stopped containers ?

See also:

- *Getting inside a container*

```
$ docker system prune
```

```
WARNING! This will remove:  
        - all stopped containers  
        - all networks not used by at least one container  
        - all dangling images  
        - all build cache  
Are you sure you want to continue? [y/N] y  
Deleted Containers:  
9a47c35465927f391fefd3faeec5b88a6926430ba7bf49160e08cfbf61d9aeab  
a1919f59bab55b472597c00051c5be57aac64e2f5d5e40deba0cbe5f9f4448ff  
49268904d59e18f3b4b33f1ff11122cc3d6cefc5dbec0a0242f20f4f2dee219f  
a061133b8ff0e07b63285573b2f3e4dc9ac598c36737d32c42ff0d80af7d5668  
  
Deleted Networks:  
ch4-message-board-app_default  
  
Deleted Images:  
deleted: sha256:e43bb6363c1ff911ce34c76475cfbc4020df989221710052a8be91f7702afcab  
deleted: sha256:46ee23e3a5a944b87b11ba03fda425d9b79a922c9df4e958def47785a5303965  
deleted: sha256:d373c573904be4c45edce0494c202f7a1cf44c87515ad24b2c2c80824b734115  
deleted: sha256:aee4f1ad67db567e681ed8847ab56c87489ab44bfd1cc183f9a75fc1164ce4a7  
deleted: sha256:724bf0a6facc9e4efd4e865c995a683e586981deb6310115269f864cda772836  
  
Total reclaimed space: 8.349kB
```

16.4 Where can I find example compose files ?

There are many examples of Compose files on github.

CHAPTER 17

Hébergeurs Docker

Contents

- *Hébergeurs Docker*
 - *Gitlab*
 - *Amazon*

17.1 Gitlab

Gitlab peut héberger des images Docker.

17.2 Amazon

See also:

- <http://www.journaldunet.com/solutions/cloud-computing/1205896-comment-aws-supporte-t-il-vraiment-docker/>

CHAPTER 18

Docker documentation

18.1 Docker aquasec documentation

See also:

- <https://www.aquasec.com/wiki>

18.1.1 About this Site

This website brings together thousands of online resources about container technology.

Containers are nothing new: as early as 1982 Unix administrators could launch isolated processes, similar to today's containers, using the chroot command.

The first modern container was probably Linux-VServer released in 2001.

Containers matured considerably in the 12 years that followed, until the rise of Docker which finally took containers to the mainstream.

Today cloud computing, deployment, DevOps and agile development are almost synonymous with containers. So much has been written on this complex subject, and few have attempted to organize this corpus into a meaningful format.

At Aqua Security, a pioneer in container security, we took upon ourselves to fill this gap and collect the most important writings about container technology - from conceptual articles and best practices to vendor information and how to guides - to help the growing community make sense of the space. The end result will include over 200 sub-topics around containers, container platforms, container orchestration and more. With a special focus on Docker and Kubernetes which are becoming ubiquitous in modern container setups.

CHAPTER 19

Docker people

19.1 Bret Fischer

See also:

- <https://twitter.com/BretFisher>
- <https://github.com/BretFisher/ama/issues>

19.1.1 News

19.1.1.1 2018

See also:

- *Only one host for production environment. What to use: docker-compose or single node swarm ?*

19.2 Nick Janetakis

See also:

- <https://github.com/nickjj>
- <https://galaxy.ansible.com/nickjj>
- <https://twitter.com/nickjanetakis>
- <https://nickjanetakis.com/blog/>



19.2.1 Best practices

See also:

- *Docker Best practices from Nick Janetakis*

19.3 Mickael Bright

See also:

- <https://github.com/mjbright>
- <https://medium.com/@mjbrightfr>
- <https://twitter.com/mjbright>
- <https://mjbright.github.io/Talks/>
- <https://mjbright.blogspot.com>

Contents

- *Mickael Bright*
 - *Activités septembre 2018 à Grenoble*

19.3.1 Activités septembre 2018 à Grenoble

See also:

- [Kubernetes news 2018-09](#)

19.4 Stéphane Beuret

See also:

- <https://twitter.com/Saphoooo>
- https://connect.ed-diamond.com/auteur/view/73156-beuret_stephane
- <https://github.com/de13>
- <https://www.meetup.com/fr-FR/Luxembourg-Rancher-Meetup/members/216544162/>

Contents

- *Stéphane Beuret*
 - *Activités 2018*
 - * *GNU/Linux Mag hors série N°98 septembre 2018*
 - * *GNU/linux mag N°217 juillet 2018*
 - * *GNU/linux mag N°214 avril 2018*



- * *GNU/linux mag N°211 janvier 2018*
- * *GNU/linux mag N°204 mai 2017*

19.4.1 Activités 2018

19.4.1.1 GNU/Linux Mag hors série N°98 septembre 2018

See also:

- Mardi 4 septembre 2018 : L'édito du hors-série N°98 spécial conteneurs !
- <https://boutique.ed-diamond.com/en-kiosque/1356-gnulinux-magazine-hs-98.html>

19.4.1.2 GNU/linux mag N°217 juillet 2018

See also:

- <https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-217/Vous-avez-dit-event-driven>

19.4.1.3 GNU/linux mag N°214 avril 2018

See also:

- <https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-214/Stockage-persistant-dans-Kubernetes-avec-Rook>

19.4.1.4 GNU/linux mag N°211 janvier 2018

See also:

- <https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-211/Introduction-Serverless-et-Function-as-a-Service-FaaS>

19.4.1.5 GNU/linux mag N°204 mai 2017

See also:

- <https://connect.ed-diamond.com/GNU-Linux-Magazine/GLMF-204/Deployez-Kubernetes-sur-vos-Raspberry-Pi-avec-Kubeadm>

CHAPTER 20

linux techniques

Contents

- *linux techniques*
 - *namespaces*
 - *cgroups*

20.1 namespaces

20.2 cgroups

CHAPTER 21

Docker videos

See also:

- <https://docs.docker.com/docker-for-windows/>

21.1 2018

See also:

- <https://www.youtube.com/watch?v=YFl2mCHdv24>

CHAPTER 22

Actions/news

22.1 Actions/news 2018

22.1.1 Actions/news 2018-11

22.1.1.1 Jeudi 8 novembre 2018 : sortie de Docker 18.09

See also:

- [18.09-ce \(2018-11-08\)](#)

22.1.2 Actions/news 2018-09

Contents

- [Actions/news 2018-09](#)
 - [Démonstration de Mickael Bright à Grenoble](#)

22.1.2.1 Démonstration de Mickael Bright à Grenoble

See also:

- [Kubernetes news 2018-09](#)

22.1.3 Actions/news 2018-08

Contents

- *Actions/news 2018-08*
 - *~jpetazzo/Dérisquer son infrastructure avec les conteneurs*

22.1.3.1 ~jpetazzo/Dérisquer son infrastructure avec les conteneurs

See also:

- <https://jpetazzo.github.io/2018/08/01/containers-cloud-immutable-infrastructure-orchestration/>

Liens

- <https://code.fb.com/web/rapid-release-at-massive-scale/>

22.1.4 Actions/news 2018-06

Contents

- *Actions/news 2018-06*
 - *Et je suis passé à https avec Docker et Traefik (https, letsencrypt)*
 - * *Conclusion*
 - *Interesting Dockerfile and docker-compose files*
 - * *Joe Jasinski*
 - * *Jeff Triplett*
 - * *cookiecutter-django docker-postgres backups*
 - *Introducing an Easier Way To Design Applications in Docker Desktop*
 - *Docker adoption*

22.1.4.1 Et je suis passé à https avec Docker et Traefik (https, letsencrypt)

See also:

- <https://www.it-wars.com/posts/performance/docker-traefik-letsencrypt/>

22.1.4.1.1 Conclusion

J'utilise Traefik depuis quelques temps déjà pour la partie Docker de mon infrastructure, son support natif de Letsecnrypt pour passer mon site web existant en https/http2, a été réalisé en moins de 10 min.

Je mesure un gain de performance et je suis maintenant tranquille par rapport à la politique de Google concernant les sites non-https !

22.1.4.2 Interesting Dockerfile and docker-compose files

22.1.4.2.1 Joe Jasinski

See also:

- <https://github.com/JoeJasinski/docker-django-demo/blob/blogpost/docker-compose.yml>
- <https://github.com/JoeJasinski/docker-django-demo/blob/blogpost/Dockerfile>
- <https://github.com/JoeJasinski/docker-django-demo/blob/blogpost/dddemo/settings/base.py>

22.1.4.2.2 Jeff Triplett

See also:

- https://raw.githubusercontent.com/jefftriplett/django-startproject/master/project_template/Dockerfile
- https://github.com/jefftriplett/django-startproject/blob/master/project_template/docker-compose.yml
- https://github.com/jefftriplett/django-startproject/blob/master/project_template/config/settings.py-tpl

22.1.4.2.3 cookiecutter-django docker-postgres backups

See also:

- <https://cookiecutter-django.readthedocs.io/en/latest/index.html>
- <https://cookiecutter-django.readthedocs.io/en/latest/developing-locally-docker.html>
- <https://cookiecutter-django.readthedocs.io/en/latest/docker-postgres-backups.html>
- <https://cookiecutter-django.readthedocs.io/en/latest/deployment-with-docker.html>

22.1.4.3 Introducing an Easier Way To Design Applications in Docker Desktop

See also:

- <https://blog.docker.com/2018/06/design-applications-in-docker-desktop/>

22.1.4.4 Docker adoption

See also:

- <https://www.datadoghq.com/docker-adoption/>

22.1.5 Actions/news mai 2018

Contents

- *Actions/news mai 2018*
 - *Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron*
 - *DjangoCon 2018 - An Intro to Docker for Djangonauts by Lacey Williams*

- *hard-multi-tenancy-in-kubernetes*
- *containers-security-and-echo-chambers*
- *Aly Sivji, Joe Jasinski, tathagata dasgupta (t) - Docker for Data Science - PyCon 2018*
 - * *Description*
- *Créez un cluster hybride ARM/AMD64 (GNU/Linux N°215 mai 2018)*

22.1.5.1 Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron

See also:

- *Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron*

22.1.5.2 DjangoCon 2018 - An Intro to Docker for Djangonauts by Lacey Williams

See also:

- <https://www.youtube.com/watch?v=v5jfDDg55xs&feature=youtu.be&a=>
- *A Brief Intro to Docker for Djangonauts par Lacey Williams*

22.1.5.3 hard-multi-tenancy-in-kubernetes

See also:

- <https://blog.jessfraz.com/post/hard-multi-tenancy-in-kubernetes/>

22.1.5.4 containers-security-and-echo-chambers

See also:

- <https://blog.jessfraz.com/post/containers-security-and-echo-chambers/>

22.1.5.5 Aly Sivji, Joe Jasinski, tathagata dasgupta (t) - Docker for Data Science - PyCon 2018

See also:

- <https://github.com/docker-for-data-science/docker-for-data-science-tutorial>
- <https://www.youtube.com/watch?v=jbb1dbFaovg>
- <https://t.co/ZW7g1JY3va>

22.1.5.5.1 Description

Jupyter notebooks simplify the process of developing and sharing Data Science projects across groups and organizations. However, when we want to deploy our work into production, we need to extract the model from the notebook and package it up with the required artifacts (data, dependencies, configurations, etc) to ensure it works in other environments.

Containerization technologies such as Docker can be used to streamline this workflow.

This hands-on tutorial presents Docker in the context of Reproducible Data Science - from idea to application deployment.

You will get a thorough introduction to the world of containers; learn how to incorporate Docker into various Data Science projects; and walk through the process of building a Machine Learning model in Jupyter and deploying it as a containerized Flask REST API.

22.1.5.6 Créez un cluster hybride ARM/AMD64 (GNU/Linux N°215 mai 2018)

22.1.6 Actions/news avril 2018

Contents

- *Actions/news avril 2018*
 - *Les slides de Petazzoni pour les formations docker et kubernetes d'avril 2018*
 - * *Le répertoire source des slides*
 - * *Autres conférences passées et futures*
 - *Docker for the busy researcher (from Erik Matsen)*
 - * *Why Docker ?*

22.1.6.1 Les slides de Petazzoni pour les formations docker et kubernetes d'avril 2018

- <https://avril2018.container.training/>
- <https://avril2018.container.training/intro.yml.html#1> (Introduction to containers, 662 slides)
- <https://avril2018.container.training/kube.yml.html#1> (introduction to orchestration with kubernetes, 384 slides)

22.1.6.1.1 Le répertoire source des slides

- <https://github.com/jpetazzo/container.training>

22.1.6.1.2 Autres conférences passées et futures

- <http://container.training/>

22.1.6.2 Docker for the busy researcher (from Erik Matsen)

See also:

- <http://erick.matsen.org/2018/04/19/docker.html>

22.1.6.2.1 Why Docker ?

Have you ever been frustrated because a software package's installation instructions were incomplete ? Or have you wanted to try out software without going through a complex installation process? Or have you wanted to execute your software on some remote machine in a defined environment?

Docker can help.

In my group, we use Docker to make sure that our code compiles properly in a defined environment and analyses are reproducible. We automatically create Docker images through Dockerfiles. This provides a clear list of dependencies which are guaranteed to work starting from a defined starting point.

Once a Docker image is built, it can be run anywhere that runs the Docker engine.

22.1.7 Actions/news mars 2018

22.1.7.1 Jeudi 29 mars 2018 : Running Your Own Registry

See also:

- <https://blog.sixeyed.com/windows-weekly-dockerfile-20-running-your-own-registry/>

22.1.7.1.1 Docker Registry

The registry is the “ship” part of the build, ship, run workflow.

You package your app in a Docker image using a Dockerfile and docker image build, and the output is an image on your machine (or the CI server that ran the build).

22.1.7.2 Jeudi 29 mars 2018 : Article de Jérôme Petazzoni : Containers par où commencer ?

See also:

- <https://jpetazzo.github.io/2018/03/28/containers-par-ou-commencer/>

22.1.8 Actions/news février 2018

22.1.8.1 Mardi 13 février 2018: import d'une nouvelle base de données données db_id3_intranet

Contents

- *Mardi 13 février 2018: import d'une nouvelle base de données données db_id3_intranet*
 - *Suppression du volume djangoid3_intranet_volume (docker volume rm djangoid3_intranet_volume)*
 - *Import de la nouvelle base de données (docker-compose -f docker-compose_for_existing_database.yml up -build)*
 - *Accès à la nouvelle base de données (docker-compose exec db bash)*
 - *Arrêt du service (docker-compose -f .docker-compose_for_existing_database.yml down)*

22.1.8.1.1 Suppression du volume djangoid3_intranet_volume (docker volume rm djangoid3_intranet_volume)

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\django_
→id3> docker volume ls
```

DRIVER	VOLUME NAME
local	djangoid3_intranet_volume
local	postgresql_volume_intranet

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\django_
→id3> docker volume rm djangoid3_intranet_volume
```

djangoid3_intranet_volume

22.1.8.1.2 Import de la nouvelle base de données (docker-compose -f docker-compose_for_existing_database.yml up --build)

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\django_
→id3> docker-compose -f docker-compose_for_existing_database.yml up --build
```

```
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. →
→All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "djangoid3_default" with the default driver
Creating volume "djangoid3_intranet_volume" with default driver
Building db
Step 1/3 : FROM postgres:10.2
--> 6e3b6a866c37
Step 2/3 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias →
→fr_FR.UTF-8
--> Using cache
--> 65da73d90928
Step 3/3 : ENV LANG fr_FR.utf8
--> Using cache
--> a932c8fcf807
Successfully built a932c8fcf807
Successfully tagged djangoid3_db:latest
Creating container_database ... done
Attaching to container_database
container_database | Les fichiers de ce cluster appartiendront à l'utilisateur →
→postgres .
container_database | Le processus serveur doit également lui appartenir.
container_database |
container_database | L'instance sera initialisée avec la locale → fr_FR.utf8 .
container_database | L'encodage par défaut des bases de données a été configuré →
→en conséquence
container_database | avec → UTF8 .
container_database | La configuration de la recherche plein texte a été →
→initialisée à → french .
```

(continues on next page)

(continued from previous page)

```

container_database |
container_database | Les sommes de contrôles des pages de données sont desactivées.
→désactivées.
container_database |
container_database | correction des droits sur le répertoire existant /var/lib/
→postgresql/data... ok
container_database | création des sous-répertoires... ok
container_database | sélection de la valeur par défaut de max_connections... 100
container_database | sélection de la valeur par défaut pour shared_buffers...128MB
container_database | sélection de l'implémentation de la mémoire partagéedynamique...posix
container_database | création des fichiers de configuration... ok
container_database | lancement du script bootstrap...ok
container_database | exécution de l'initialisation après bootstrap...ok
container_database | synchronisation des données sur disqueok
container_database |
container_database | ATTENTION : active l'authentification à trust pour lesconnexions
→connexions
container_database | locales.
container_database | Vous pouvez changer cette configuration en éditant lefichier pg_hba.conf
→fichier pg_hba.conf
container_database | ou en utilisant l'option -A, ou --auth-local et --auth-hostau prochain
→au prochain
container_database | lancement d'initdb.
container_database |
container_database | Succès. Vous pouvez maintenant lancer le serveur de bases dedonnées en utilisant :
→données en utilisant :
container_database |
container_database |      pg_ctl -D /var/lib/postgresql/data -l fichier de tracestart
→start
container_database |
container_database | ****
container_database | WARNING: No password has been set for the database.
container_database | This will allow anyone with access to the
container_database | Postgres port to access your database. In
container_database | Docker's default configuration, this is
container_database | effectively any other container on the same
container_database | system.
container_database |
container_database |      Use "-e POSTGRES_PASSWORD=password" to set
container_database | it in "docker run".
container_database | ****
container_database | en attente du démarrage du serveur....2018-02-14 12:52:43.
→323 UTC [38] LOG: en écoute sur IPv4, adresse à 127.0.0.1 , port 5432
container_database | 2018-02-14 12:52:43.342 UTC [38] LOG: n'a pas pu lier IPv6à l'adresse à ::1 : Ne peut attribuer l'adresse demandée
→à l'adresse à ::1 : Ne peut attribuer l'adresse demandée
container_database | 2018-02-14 12:52:43.342 UTC [38] ASTUCE : Un autrepostmaster fonctionne-t'il déjà sur le port 5432 ?
→postmaster fonctionne-t'il déjà sur le port 5432 ?
container_database | Sinon, attendez quelques secondes et réessayez.
container_database | 2018-02-14 12:52:43.508 UTC [38] LOG: écoute sur la socketUnix à /var/run/postgresql/.s.PGSQL.5432
→Unix à /var/run/postgresql/.s.PGSQL.5432
container_database | 2018-02-14 12:52:43.693 UTC [39] LOG: le système de basesde données a été arrêté à 2018-02-14 12:52:40 UTC
→de données a été arrêté à 2018-02-14 12:52:40 UTC
container_database | 2018-02-14 12:52:43.791 UTC [38] LOG: le système de basesde données est prêt pour accepter les connexions
→de données est prêt pour accepter les connexions
container_database | effectué

```

(continues on next page)

(continued from previous page)

```

container_database | serveur démarré
container_database | ALTER ROLE
container_database |
container_database |
container_database | /usr/local/bin/docker-entrypoint.sh: running /docker-
→entrypoint-initdb.d/dump_id3_intranet.sql
container_database | CREATE ROLE
container_database | SET
container_database | SET
container_database | SET

...
container_database | ALTER TABLE
container_database | ALTER TABLE
container_database | ALTER TABLE
container_database | GRANT
container_database |
container_database |
container_database | en attente de l'arrêt du serveur....2018-02-14 12:53:39.199
→UTC [38] LOG: a reçu une demande d'arrêt rapide
    container_database | 2018-02-14 12:53:39.297 UTC [38] LOG: annulation des
→transactions actives
    container_database | 2018-02-14 12:53:39.302 UTC [38] LOG: processus de travail:
→logical replication launcher (PID 45) quitte avec le code de sortie 1
    container_database | 2018-02-14 12:53:39.304 UTC [40] LOG: arrêt en cours
    container_database | .....2018-02-14 12:53:46.826 UTC [38] LOG: le système de
→base de données est arrêté
    container_database | effectué
    container_database | serveur arrêté
    container_database |
    container_database | PostgreSQL init process complete; ready for start up.
    container_database |
    container_database | 2018-02-14 12:53:47.027 UTC [1] LOG: en écoute sur IPv4,
→adresse % 0.0.0.0 , port 5432
    container_database | 2018-02-14 12:53:47.027 UTC [1] LOG: en écoute sur IPv6,
→adresse % :: , port 5432
    container_database | 2018-02-14 12:53:47.252 UTC [1] LOG: écoute sur la socket
→Unix % /var/run/postgresql/.s.PGSQL.5432
    container_database | 2018-02-14 12:53:47.522 UTC [68] LOG: le système de bases
→de données a été arrêté à 2018-02-14 12:53:46 UTC
    container_database | 2018-02-14 12:53:47.648 UTC [1] LOG: le système de bases
→de données est prêt pour accepter les connexions

```

22.1.8.1.3 Accès à la nouvelle base de données (docker-compose exec db bash)

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\django_
→id3> docker-compose exec db bash
```

```
root@365f7c4e3096:/# psql -U postgres
```

```
psql (10.2 (Debian 10.2-1.pgdg90+1))
Saisissez « help » pour l'aide.
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure with files like `db.dump_2018_02_14.sql`, `dump_id3_intranet.sql`, and `docker-compose_for_existing_database.yml`.
- Code Editor:** Displays the `docker-compose_for_existing_database.yml` file content:

```

version: "3"
services:
  db:
    build:
      context: .
      dockerfile: db/Dockerfile
      container_name: container_database
    ports:
      - 5432:5432
    volumes:
      - intranet_volume:/var/lib/postgresql/data
      # First import of the database
      - ./init:/docker-entrypoint-initdb.d/
  volumes:
    intranet_volume:

```

- Terminal:** Shows the following command and its output in a Windows PowerShell window:

```

PS Y:\projects_id3\PSN001\XLOGICA135_tutorial_docker\tutorial_docker\tutoriels\django_id3> docker-compose exec db bash
root@365f7c4e3096:/# psql -U postgres
psql (10.2 (Debian 10.2.1-0.pgdg90+1))
Saisissez « help » pour l'aide.

postgres=# \l
           Liste des bases de données
   Nom   | Propriétaire | Encodage | Collationnement | Type caract. |   Droits d'accès
---+---+---+---+---+---+
db_id3_intranet | id3admin | UTF8 | fr_FR.UTF-8 | fr_FR.UTF-8 |
postgres | postgres | UTF8 | fr_FR.utf8 | fr_FR.utf8 | =c/postgres +
template0 | postgres | UTF8 | fr_FR.utf8 | fr_FR.utf8 | postgres=CTc/postgres +
template1 | postgres | UTF8 | fr_FR.utf8 | fr_FR.utf8 | =c/postgres +
(4 lignes)

postgres=# \c db_id3_intranet
Vous êtes maintenant connecté à la base de données « db_id3_intranet » en tant qu'utilisateur « postgres ».
db_id3_intranet=# \dt
           Liste des relations
   Schéma |   Nom   | Type | Propriétaire
---+---+---+---+

```

Fig. 1: Accès à la base de données mise à jour avec les données de sybase

```
postgres=# \l
```

Liste des bases de données					
Nom	Propriétaire	Encodage	Collationnement	Type caract.	
Droits d'accès					
db_id3_intranet	id3admin	UTF8	fr_FR.UTF-8	fr_FR.UTF-8	
postgres	postgres	UTF8	fr_FR.utf8	fr_FR.utf8	
template0	postgres	UTF8	fr_FR.utf8	fr_FR.utf8	=c/
postgres	+				
template1	postgres	UTF8	fr_FR.utf8	fr_FR.utf8	=c/
postgres	+				
(4 lignes)					

```
postgres=# \c db_id3_intranet
```

Vous êtes maintenant connecté à la base de données « db_id3_intranet » en tant qu'utilisateur « postgres ».

```
db_id3_intranet=# \dt
```

22.1.8.1.4 Arrêt du service (`docker-compose -f .\docker-compose_for_existing_database.yml down`)

```
docker-compose -f .\docker-compose_for_existing_database.yml down
```

22.1.8.2 Mardi 13 février 2018: mise en place d'une base de données PostgreSQL 10.2 avec import de la base de données db_id3_intranet

Contents

- *Mardi 13 février 2018: mise en place d'une base de données PostgreSQL 10.2 avec import de la base de données db_id3_intranet*
 - *docker-compose_for_existing_database.yml*
 - *Contenu du répertoire init*
 - * *Création de la base db_id3_intranet*
 - * *Création de l'utilisateur id3admin*

22.1.8.2.1 docker-compose_for_existing_database.yml

La ligne très importante qu'il fallait trouver est la ligne:

```
- ./init:/docker-entrypoint-initdb.d/
```

```
# docker-compose_for_existing_database.yml
# Create a new persistant intranet_volume from init/db.dump_2018_02_01.sql
version: "3"
services:
  db:
    build:
      context: .
      dockerfile: db/Dockerfile
    container_name: container_database
    ports:
      # the 5432 host port is occupied by a local postgresql server
      - 5433:5432
    volumes:
      - intranet_volume:/var/lib/postgresql/data
      # First import of the database
      - ./init:/docker-entrypoint-initdb.d/
volumes:
  intranet_volume:
```

22.1.8.2.2 Contenu du répertoire init

Mode	LastWriteTime	Length	Name
-a---	13/02/2018 11:05	34177687	db.dump_2018_02_01.sql

L'entête du fichier SQL étant:

```
-- 
-- PostgreSQL database dump
-- 

-- Dumped from database version 10.1
-- Dumped by pg_dump version 10.1

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

-- 
-- Name: db_id3_intranet; Type: DATABASE; Schema: -; Owner: id3admin
-- 

CREATE DATABASE db_id3_intranet WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_
→COLLATE = 'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

CREATE USER id3admin WITH
```

(continues on next page)

(continued from previous page)

```

LOGIN
NOSUPERUSER
INHERIT
NOCREATEDB
NOCREATEROLE
NOREPLICATION
password 'id338';

ALTER DATABASE db_id3_intranet OWNER TO id3admin;

\connect db_id3_intranet

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

-- 
-- Name: db_id3_intranet; Type: COMMENT; Schema: -; Owner: id3admin
-- 

COMMENT ON DATABASE db_id3_intranet IS 'La base db_id3_intranet';

```

Création de la base db_id3_intranet

```

CREATE DATABASE db_id3_intranet WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_
↪COLLATE = 'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

```

Création de l'utilisateur id3admin

```

CREATE USER id3admin WITH
  LOGIN
  NOSUPERUSER
  INHERIT
  NOCREATEDB
  NOCREATEROLE
  NOROLE
  password 'id338';

```

22.1.8.3 Lundi 12 février 2018: mise en place d'une base de données PostgreSQL 10.2

Contents

- *Lundi 12 février 2018: mise en place d'une base de données PostgreSQL 10.2*
 - *Dockerfile*

- *docker-compose.yml*
- *Accès HeidiSQL à partir de la machine hôte*

22.1.8.3.1 Dockerfile

```
# https://store.docker.com/images/postgres
FROM postgres:10.2
# avec cette image on peut mettre en place la locale fr_FR.utf8
RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8
```

22.1.8.3.2 docker-compose.yml

```
version: "3"
services:
  db:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      # the 5432 host port is occupied by a local postgresql server
      - 5433:5432
    volumes:
      - volume_intranet:/var/lib/postgresql/data/
volumes:
  volume_intranet:
```

22.1.8.3.3 Accès HeidiSQL à partir de la machine hôte

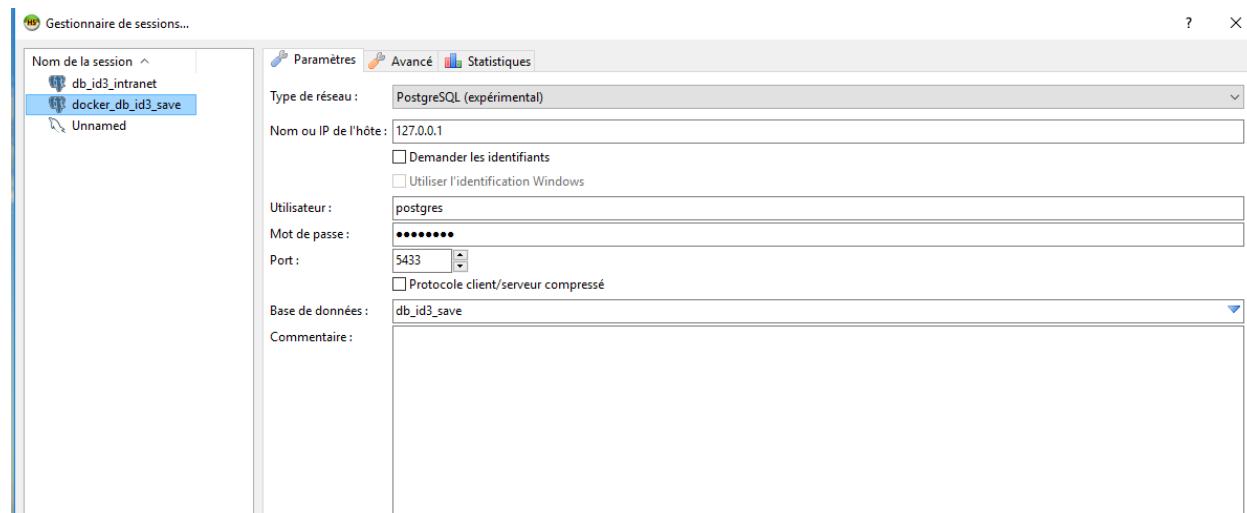


Fig. 2: Accès HeidiSQL à partir de la machine hôte sur le port 5433

22.1.9 Actions/news janvier 2018

22.1.9.1 Mercredi 31 janvier 2018 : export/import d'une base de données PostgreSQL (tutoriel PostgreSQL)

See also:

- *Mercredi 31 janvier 2018 : export/import d'une base de données PostgreSQL (tutoriel PostgreSQL)*

22.1.9.1.1 Dockerfile

```
FROM postgres:10.1
RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8
```

22.1.9.1.2 docker-compose.yml

```
version: "3"
services:
  db:
    build:
      context: .
      dockerfile: Dockerfile
    image: postgres:10.1
    container_name: container_intranet
    volumes:
      - volume_intranet:/var/lib/postgresql/data/
      - .:/code

volumes:
  volume_intranet:
```

22.1.9.1.3 Export

- pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet

22.1.9.1.4 Import

- psql -U postgres -f db.dump.sql

22.1.9.1.5 Commandes docker-compose

- docker-compose up
- docker-compose down
- docker-compose exec db bash

22.1.9.2 Mercredi 31 janvier 2018 : Bilan mardi 30 janvier 2018

See also:

- [Tutoriel Docker et Postgresql](#)
- [Mardi 30 janvier 2018 : écriture des fichiers Dockerfile et docker-compose.yml](#)
- [Images PostgreSQL](#)

Contents

- [Mercredi 31 janvier 2018 : Bilan mardi 30 janvier 2018](#)
 - [Suppression de la base db_id3_intranet](#)
 - * `psql -U postgres`
 - * `l`
 - * `drop database db_id3_intranet;`
 - [Bilan mardi 30 janvier 2018](#)
 - [Pour lancer PostgreSQL](#)
 - [Pour accéder au conteneur](#)
 - * `docker ps`
 - * `docker exec -ti caa4db30ee94 bash`
 - [Livre PostgreSQL : Administration et exploitation de vos bases de données](#)

22.1.9.2.1 Suppression de la base db_id3_intranet

`psql -U postgres`

```
root@caa4db30ee94:/ # psql -U postgres
```

```
psql (10.1)
Type "help" for help.
```

|

```
postgres=# \l
```

List of databases						
Name	Owner	Encoding	Collate	Ctype	Access	
db_id3_intranet	id3admin	UTF8	en_US.utf8	en_US.utf8		
postgres	postgres	UTF8	en_US.utf8	en_US.utf8		

(continues on next page)

(continued from previous page)

template0		postgres		UTF8		en_US.utf8		en_US.utf8		=c/postgres	✉
↳ +											✉
											✉
↳ postgres=CTc/postgres											✉
template1		postgres		UTF8		en_US.utf8		en_US.utf8		=c/postgres	✉
↳ +											✉
											✉
↳ postgres=CTc/postgres											✉
(4 rows)											

drop database db_id3_intranet;

postgres=# drop database db_id3_intranet;

DROP DATABASE

22.1.9.2.2 Bilan mardi 30 janvier 2018

Pour pouvoir importer une base données PostgreSQL, il faut utiliser cette suite de commandes dans le fichier docker-compose.yml.

```
version: "3"

services:
  db:
    image: postgres:10.1
    container_name: container_intranet
    volumes:
      - volume_intranet:/var/lib/postgresql/data/
      - ./code

volumes:
  volume_intranet:
```

La commande `./code` permet de voir ce qu'il y a dans le répertoire du coté *host*.

root@caa4db30ee94:/# ls -als code

```
total 33897
4 drwxr-xr-x 2 root root 4096 Jan 31 08:24 .
4 drwxr-xr-x 1 root root 4096 Jan 30 13:46 ..
33776 -rwxr-xr-x 1 root root 34586512 Jan 25 13:51 db_id3_intranet_2018_01_25.sql
1 -rwxr-xr-x 1 root root 214 Jan 30 13:46 docker-compose.yml
24 -rwxr-xr-x 1 root root 23949 Jan 30 14:04 postgresql.rst
8 -rwxr-xr-x 1 root root 6238 Jan 31 08:24 README.txt
80 -rwxr-xr-x 1 root root 80802 Jan 22 12:03 stack_overflow_postgres.png
```

On voit bien le fichier `db_id3_intranet_2018_01_25.sql`

22.1.9.2.3 Pour lancer PostgreSQL

```
docker-compose up
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql> docker-compose up
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Starting container_intranet ... done
Attaching to container_intranet
container_intranet 2018-01-31 07:46:47.402 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
container_intranet 2018-01-31 07:46:47.402 UTC [1] LOG: listening on IPv6 address "::", port 5432
container_intranet 2018-01-31 07:46:47.556 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
container_intranet 2018-01-31 07:46:47.786 UTC [20] LOG: database system was interrupted; last known up at 2018-01-30 14:43:18 UTC
container_intranet 2018-01-31 07:46:49.498 UTC [20] LOG: database system was not properly shut down; automatic recovery in progress
container_intranet 2018-01-31 07:46:49.688 UTC [20] LOG: redo starts at 0/167C828
container_intranet 2018-01-31 07:46:49.688 UTC [20] LOG: invalid record length at 0/167c908: wanted 24, got 0
container_intranet 2018-01-31 07:46:49.688 UTC [20] LOG: redo done at 0/167C8D0
container_intranet 2018-01-31 07:46:50.423 UTC [1] LOG: database system is ready to accept connections
```

Fig. 3: docker-compose up

22.1.9.2.4 Pour accéder au conteneur

docker ps

```
docker ps
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	
Status	Ports	Names		
caa4db30ee94	postgres:10.1	"docker-entrypoint.s..."	19 hours ago	
Up 34 minutes	5432/tcp	container_intranet		

docker exec -ti caa4db30ee94 bash

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker exec -ti caa4db30ee94 bash
```

```
root@caa4db30ee94:/# ls -als
total 80
4 drwxr-xr-x 1 root root 4096 Jan 30 13:46 .
4 drwxr-xr-x 1 root root 4096 Jan 30 13:46 ..
4 drwxr-xr-x 1 root root 4096 Dec 12 06:04 bin
4 drwxr-xr-x 2 root root 4096 Nov 19 15:25 boot
4 drwxr-xr-x 2 root root 4096 Jan 31 08:22 code
0 drwxr-xr-x 5 root root 340 Jan 31 07:46 dev
4 drwxr-xr-x 2 root root 4096 Dec 12 06:04 docker-entrypoint-initdb.d
0 lrwxrwxrwx 1 root root 34 Dec 12 06:05 docker-entrypoint.sh -> usr/local/bin/
→docker-entrypoint.sh
0 -rwxr-xr-x 1 root root 0 Jan 30 13:46 .dockercfg
4 drwxr-xr-x 1 root root 4096 Jan 30 13:46 etc
4 drwxr-xr-x 2 root root 4096 Nov 19 15:25 home
4 drwxr-xr-x 1 root root 4096 Dec 10 00:00 lib
```

(continues on next page)

(continued from previous page)

```

4 drwxr-xr-x  2 root root 4096 Dec 10 00:00 lib64
4 drwxr-xr-x  2 root root 4096 Dec 10 00:00 media
4 drwxr-xr-x  2 root root 4096 Dec 10 00:00 mnt
4 drwxr-xr-x  2 root root 4096 Dec 10 00:00 opt
0 dr-xr-xr-x 132 root root    0 Jan 31 07:46 proc
4 drwx----- 1 root root 4096 Jan 30 14:32 root
4 drwxr-xr-x  1 root root 4096 Dec 12 06:05 run
4 drwxr-xr-x  1 root root 4096 Dec 12 06:04 sbin
4 drwxr-xr-x  2 root root 4096 Dec 10 00:00 srv
0 dr-xr-xr-x 13 root root    0 Jan 31 07:46 sys
4 drwxrwxrwt  1 root root 4096 Jan 30 13:46 tmp
4 drwxr-xr-x  1 root root 4096 Dec 10 00:00 usr
4 drwxr-xr-x  1 root root 4096 Dec 10 00:00 var

```

22.1.9.2.5 Livre PostgreSQL : Administration et exploitation de vos bases de données

De nombreuses informations **très intéressantes**.

- psql -f nom_fichier.sql
- explications sur les bases template0 et template1

22.1.9.3 Mardi 30 janvier 2018 : écriture des fichiers Dockerfile et docker-compose.yml

See also:

- *Tutoriel Docker et Postgresql*
- *Mercredi 31 janvier 2018 : Bilan mardi 30 janvier 2018*

Contents

- *Mardi 30 janvier 2018 : écriture des fichiers Dockerfile et docker-compose.yml*
 - *Objectifs pour la journée*
 - *Avancement, découverte*
 - *Historique*

22.1.9.3.1 Objectifs pour la journée

Mises et point et premières exécutions.

Dans un premier temps on ne prend pas en charge les secrets.

22.1.9.3.2 Avancement, découverte

- je repasse sur le tutoriel *postgresql* pour essayer de comprendre les volumes.

22.1.9.3.3 Historique

- ajout MISC95

```
CREATE DATABASE db_test WITH OWNER = id3admin ENCODING = 'UTF8' CONNECTION LIMIT = -1;

C:\Tmp>psql -U postgres < create_database.sql
Mot de passe pour l'utilisateur postgres : id338

CREATE DATABASE
```

22.1.9.4 Lundi 29 janvier 2018 : encore un nouveau tutoriel : A Simple Recipe for Django Development In Docker (Bonus: Testing with Selenium) de Jacob Cook

See also:

- <https://medium.com/@adamzeitcode/a-simple-recipe-for-django-development-in-docker-bonus-testing-with-selenium-6a038ec1>

22.1.9.4.1 Analyse et plan de travail pour la journée

S'inspirer des 4 tutoriels pour créer les fichiers Dockerfile et Docker-compose.yml

- <https://medium.com/@adamzeitcode/a-simple-recipe-for-django-development-in-docker-bonus-testing-with-selenium-6a038ec1>
- <https://peakwinter.net/blog/modern-devops-django/>
- <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>
- <https://wsvincent.com/django-docker-postgresql/>

22.1.9.4.2 Autre projet intéressant

dockerize-all-the-things

See also:

- <https://github.com/DrewDahlman/dockerize-all-the-things>

Kill it with fire

- docker rm \$(docker ps -a -q) - Kills all containers
- docker rmi \$(docker images -q) - will toast ALL of your images

Something to keep in mind is that sometimes docker containers and images can get bloated on your machine and you might have to toast everything.

The great thing about using docker like this is that you can quickly rebuild a project and get right back into working.

Also when you close a console you are not stopping the container, you always need to run docker-compose down when stopping a project, otherwise it will just keep running in the background.

docker rm \$(docker ps -a -q)

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\actions_
→news\2018\2018_01\01__2018_01_29> docker rm $(docker ps -a -q)
```

```
367ce1d9818a
c467c2469b34
7fb912b6a3e2
1746a16a91eb
6ee9dc365c9d
8ae3930ee2d6
97592a1a70ea
8fffcde2f70f6
3d1169398f02
e629ebfc3981
ddbe7a8e2502
7c1af485479
ebe371507dc2
2b8fff5f4068
cb62ace67ba4
685915373a4c
e150d0531321
7d6e93a39de5
807d38ada261
eebf7e801b96
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\actions_
→news\2018\2018_01\01__2018_01_29> docker rmi -f $(docker images -q)
```

```
Untagged: saleor_celery:latest
Untagged: saleor_web:latest
Deleted: sha256:fe40ed7484fe2f111dfdc7b8e79d3353534f29cc28c9019a0b0d6fcbb2b624ac4
Deleted: sha256:b49f310a4175a6f56d4ad302c60307c989774a7c853a6dd068cbf68fc234926c
Deleted: sha256:5601669ae105acb6a632cd7d3dd473158b25ff6f1c7d65a95b04a2c12bad713d
Deleted: sha256:bf662c677b1ec758f37dac85c90d55c0c005be7f283723f0f85deaf1e0418c1c
Deleted: sha256:08889c646f293f56cf2a4bc2087a7fe3263f745536f9dd6c0d910264b2e10361
Deleted: sha256:64b9f0663d35de8d404374e8574484d60195e55507b3a87897821bc383c1b69d
Deleted: sha256:716475184da4626198a7da0d47d14072b4bb7c96384b1c6e67eb97daecd25b25
Deleted: sha256:9deb54f781dd986aab78aaeaebeef6ed8c587837595b02f7fb8b9008eb80006d6
Deleted: sha256:bb6904496c708da82760b2ca6e3f737608180e377ba88129060318a7af311398
Deleted: sha256:bc59713a5080512001bf80eecce306b85096858601f07ce23d8e0a9233ad69d9
```

22.2 Actions/news 2017

22.2.1 Actions/news août 2017

22.2.1.1 4 août 2017 “Docker et Shorewall” par Guillaume Cheramy

See also:

- <https://www.guillaume-cheramy.fr/docker-et-shorewall/>
- https://twitter.com/cheramy_linux

22.2.1.1.1 Créer les règles shorewall pour Docker

Il faut créer dans shorewall les règles pour que les conteneurs puissent avoir accès à Internet :

CHAPTER 23

Images Docker (Store Docker, ex Hub docker)

See also:

- <https://store.docker.com/>
- <https://docs.docker.com/engine/userguide/eng-image/>
- <https://github.com/docker-library>
- <https://github.com/docker-library/official-images>
- <https://hub.docker.com/explore/>

Contents

- *Images Docker (Store Docker, ex Hub docker)*
 - *Nouveau: le docker store: https://store.docker.com/*
 - *Ancien: le hub docker https://hub.docker.com/explore/*
 - *Gitlab registry*
 - *Images OS*
 - *Images langages*
 - *Images webserver: serveurs HTTP (serveurs Web)*
 - * *Images webserver : serveurs Web + reverse proxy + load balancer*
 - *Apache HTTP Server + mod_proxy*
 - *Nginx*
 - *Images db : bases de données*
 - *Images message queue*
 - *Images outils collaboratifs*

- *Images “documentation”*
- *Images outils scientifiques*
- *Images apprentissage*

23.1 Nouveau: le docker store: <https://store.docker.com/>

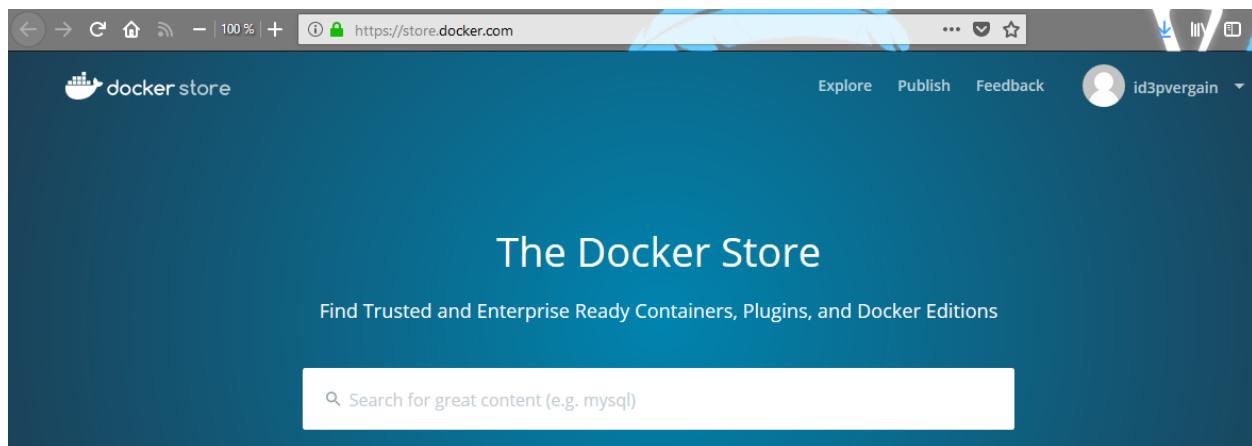


Fig. 1: <https://store.docker.com/>

23.2 Ancien: le hub docker <https://hub.docker.com/explore/>

23.3 Gitlab registry

23.3.1 GitLab Container Registry

See also:

- https://docs.gitlab.com/ce/user/project/container_registry.html

Contents

- *GitLab Container Registry*
 - *Introduction*
 - *Private registry*

23.3.1.1 Introduction

With the Docker Container Registry integrated into GitLab, **every project can have its own space to store its Docker images**.

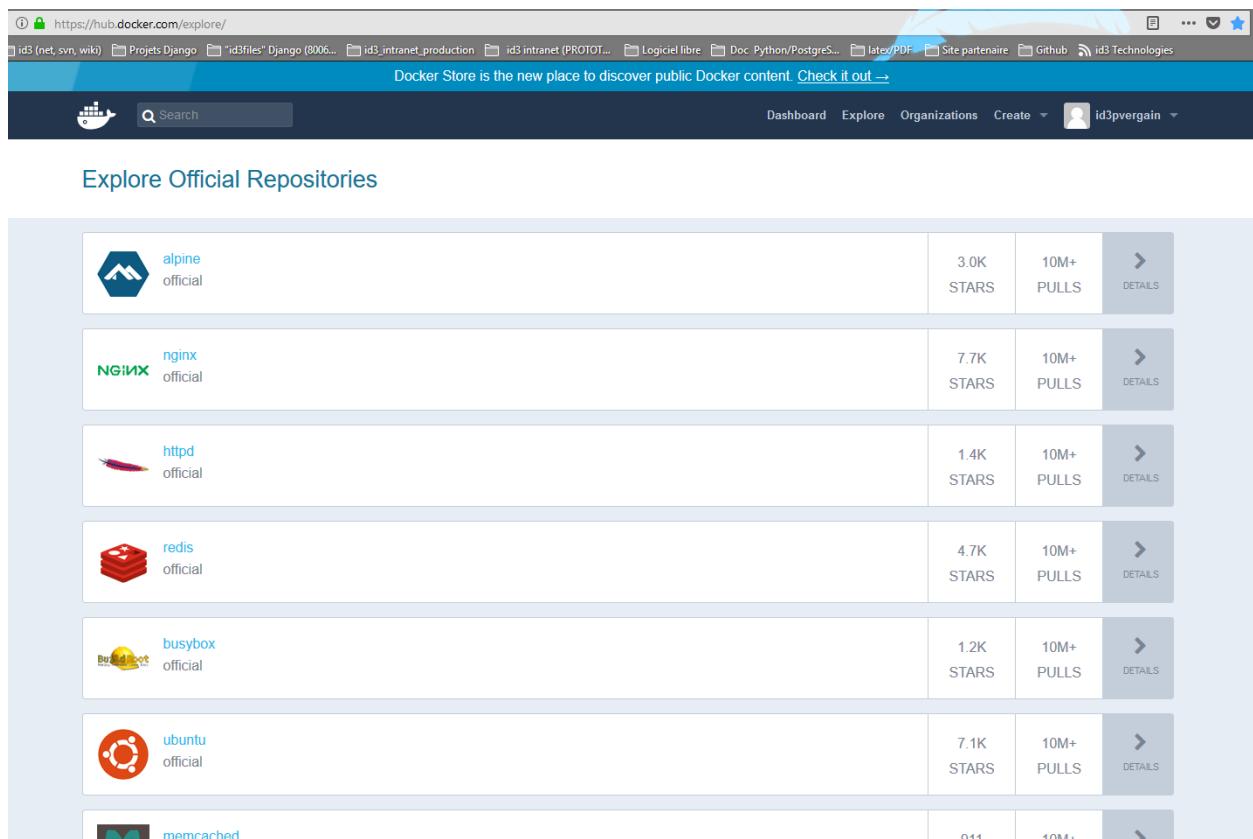


Fig. 2: <https://hub.docker.com/explore/>

23.3.1.2 Private registry

23.3.1.2.1 Private GitLab Container Registry

See also:

- https://docs.gitlab.com/ce/user/project/container_registry.html

Contents

- *Private GitLab Container Registry*
 - *Utilisation sous GNU/Linux*
 - * */etc/docker/daemon.json*
 - * *docker login*
 - * *restart docker*
 - *docker tag*
 - *docker push*

Utilisation sous GNU/Linux

/etc/docker/daemon.json

```
pvergain@UC004:~$ cat /etc/docker/daemon.json
{
  "insecure-registries" : ["dockerhub.srv.int.id3.eu:5555"]
}
```

docker login

```
docker login -u gitlab-ci-token -p "XXXXXXXXXX" "dockerhub.srv.int.id3.eu:5555"
```

restart docker

```
sudo systemctl restart docker.service
```

docker tag

```
docker tag be4f0 transactions_colombie:4_2_0_dev
```

docker push

```
docker push
```

23.4 Images OS

23.4.1 Images Alpine

See also:

- <https://store.docker.com/images/alpine>
- https://hub.docker.com/_/alpine/
- <https://www.alpinelinux.org/>
- https://fr.wikipedia.org/wiki/Alpine_Linux
- <https://github.com/gliderlabs/docker-alpine>
- <http://gliderlabs.viewdocs.io/docker-alpine/>

Contents

- *Images Alpine*
 - *Short Description*
 - *Description*
 - *Dockerfile*

Fig. 3: Le logo Alpine-linux

23.4.1.1 Short Description

A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!

23.4.1.2 Description

See also:

- https://fr.wikipedia.org/wiki/Alpine_Linux

Alpine Linux est une distribution Linux ultra-légère, orientée sécurité et basée sur Musl et BusyBox, principalement conçue pour “Utilisateur intensif qui apprécie la sécurité, la simplicité et l’efficacité des ressources”.

Elle utilise les patches PaX et Grsecurity du noyau par défaut et compile tous les binaires de l'espace utilisateur et exécutables indépendants de la position (dits “portables”) avec protection de destruction de la pile.

Cette distribution se prête particulièrement, en raison de sa légèreté, à la création d'images de containers Docker. La distribution Alpine Linux est particulièrement populaire pour cet usage .

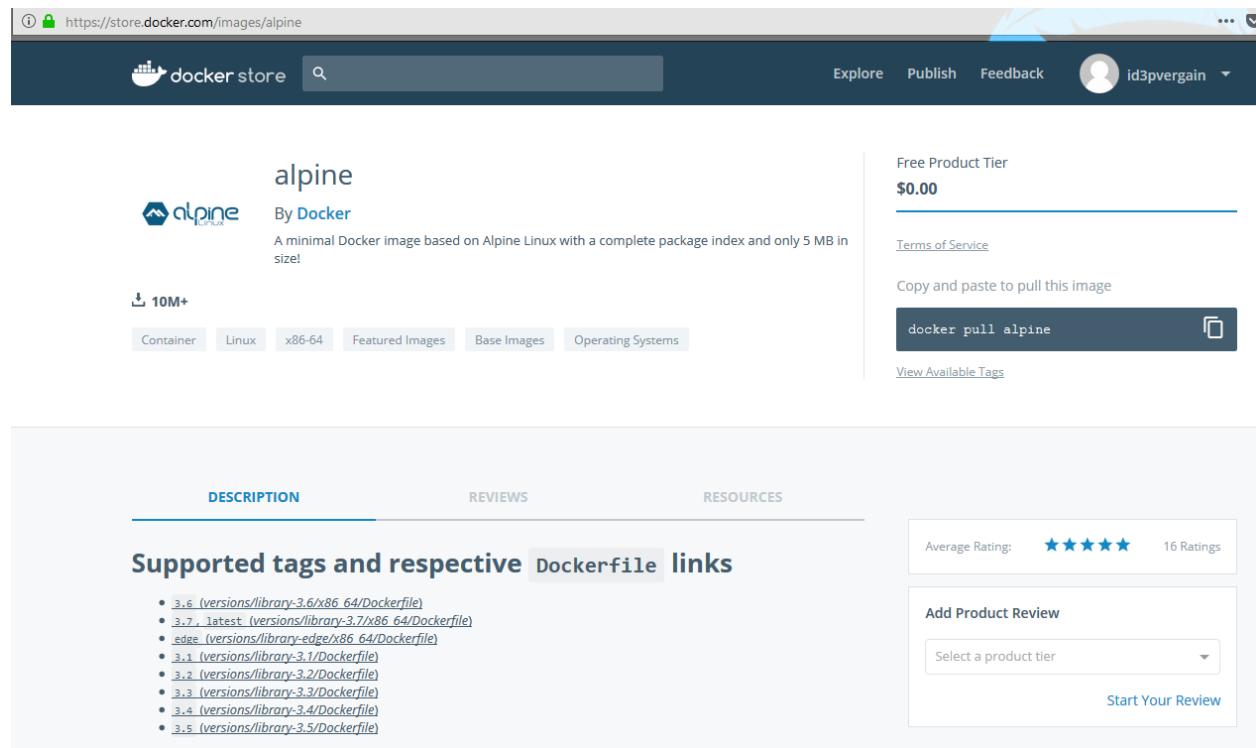


Fig. 4: <https://store.docker.com/images/alpine>

23.4.1.3 Dockerfile

```
FROM scratch
ADD rootfs.tar.xz /
CMD ["/bin/sh"]
```

23.4.2 Images Debian

See also:

- <https://store.docker.com/images/debian>
- https://hub.docker.com/_/debian/
- <https://fr.wikipedia.org/wiki/Debian>

Contents

- *Images Debian*
 - *Short Description*
 - *Description*



Fig. 5: Logo Debian

① https://hub.docker.com/r/_/debian/

Supported tags and respective Dockerfile links

- buster , buster-20171210 ([buster/Dockerfile](#))
- buster-slim ([buster/slim/Dockerfile](#))
- experimental , experimental-20171210 ([experimental/Dockerfile](#))
- jessie , jessie-20171210 , 8.10 , 8 ([jessie/Dockerfile](#))
- jessie-backports ([jessie/backports/Dockerfile](#))
- jessie-slim , 8.10-slim , 8-slim ([jessie/slim/Dockerfile](#))
- oldoldstable , oldoldstable-20171210 ([oldoldstable/Dockerfile](#))
- oldoldstable-backports ([oldoldstable/backports/Dockerfile](#))
- oldoldstable-slim ([oldoldstable/slim/Dockerfile](#))
- oldstable , oldstable-20171210 ([oldstable/Dockerfile](#))
- oldstable-backports ([oldstable/backports/Dockerfile](#))
- oldstable-slim ([oldstable/slim/Dockerfile](#))
- rc-buggy , rc-buggy-20171210 ([rc-buggy/Dockerfile](#))
- sid , sid-20171210 ([sid/Dockerfile](#))
- sid-slim ([sid/slim/Dockerfile](#))
- stable , stable-20171210 ([stable/Dockerfile](#))
- stable-backports ([stable/backports/Dockerfile](#))
- stable-slim ([stable/slim/Dockerfile](#))
- stretch , stretch-20171210 , 9.3 , 9 , latest ([stretch/Dockerfile](#))
- stretch-backports ([stretch/backports/Dockerfile](#))
- stretch-slim , 9.3-slim , 9-slim ([stretch/slim/Dockerfile](#))
- testing , testing-20171210 ([testing/Dockerfile](#))
- testing-slim ([testing/slim/Dockerfile](#))
- unstable , unstable-20171210 ([unstable/Dockerfile](#))
- unstable-slim ([unstable/slim/Dockerfile](#))
- wheezy , wheezy-20171210 , 7.11 , 7 ([wheezy/Dockerfile](#))
- wheezy-backports ([wheezy/backports/Dockerfile](#))
- wheezy-slim , 7.11-slim , 7-slim ([wheezy/slim/Dockerfile](#))

Fig. 6: https://hub.docker.com/_/debian/

23.4.2.1 Short Description

Debian is a Linux distribution that's composed entirely of free and open-source software.

23.4.2.2 Description

See also:

- <https://fr.wikipedia.org/wiki/Debian>

Debian (/de.bjan/) est une organisation communautaire et démocratique, dont le but est le développement de systèmes d'exploitation basés exclusivement sur des logiciels libres.

Chaque système, lui-même nommé Debian, réunit autour d'un noyau de système d'exploitation de nombreux éléments pouvant être développés indépendamment les uns des autres, pour plusieurs architectures matérielles. Ces éléments, programmes de base complétant le noyau et logiciels applicatifs, se présentent sous forme de « paquets » qui peuvent être installés en fonction des besoins (voir Distribution des logiciels). L'ensemble système d'exploitation plus logiciels s'appelle une distribution.

On assimile généralement ces systèmes d'exploitation au système Debian GNU/Linux, la distribution GNU/Linux de Debian, car jusqu'en 2009 c'était la seule branche parfaitement fonctionnelle. Mais d'autres distributions Debian sont en cours de développement en 2013 : Debian GNU/Hurd3, et Debian GNU/kFreeBSD5. La version Debian *Squeeze* est la première à être distribuée avec le noyau kFreeBSD en plus du noyau Linux6.

Debian est utilisée comme base de nombreuses autres distributions telles que Knoppix et Ubuntu qui rencontrent un grand succès.

23.4.3 Images Ubuntu

See also:

- <https://store.docker.com/images/ubuntu>
- https://hub.docker.com/_/ubuntu/
- [https://fr.wikipedia.org/wiki/Ubuntu_\(syst%C3%A8me_d%27exploitation\)](https://fr.wikipedia.org/wiki/Ubuntu_(syst%C3%A8me_d%27exploitation))

Contents

- *Images Ubuntu*
 - *Short Description*
 - *Description*
 - *La Philosophie d'Ubuntu*

23.4.3.1 Short Description

Ubuntu is a Debian-based Linux operating system based on free software.

23.4.3.2 Description

See also:



Fig. 7: Le logo Ubuntu

A screenshot of the Docker Hub website showing the official Ubuntu repository. The URL in the address bar is https://hub.docker.com/_/ubuntu/. The page features a blue header with the Docker logo and a search bar. Below the header, a banner announces that "ubuntu is now available in the Docker Store, the new place to discover public Docker content. Check it out →". The main content area is titled "OFFICIAL REPOSITORY" and shows the "ubuntu" repository. It includes a star icon indicating it's popular. The last push was 16 hours ago. There are two tabs: "Repo Info" (selected) and "Tags". Under "Repo Info", there are sections for "Short Description" (Ubuntu is a Debian-based Linux operating system based on free software.) and "Full Description". On the right, there is a "Docker Pull Command" section with the command "docker pull ubuntu". Below these, a list of "Supported tags and respective Dockerfile links" is provided, including: 17.10, artful-20180112, artful, rolling ([artful/Dockerfile](#)); 18.04, bionic-20171220, bionic, devel ([bionic/Dockerfile](#)); 14.04, trusty-20180112, trusty ([trusty/Dockerfile](#)); and 16.04, xenial-20180112.1, xenial, latest ([xenial/Dockerfile](#)).

Fig. 8: https://hub.docker.com/_/ubuntu/

- [https://fr.wikipedia.org/wiki/Ubuntu_\(syst%C3%A8me_d%27exploitation\)](https://fr.wikipedia.org/wiki/Ubuntu_(syst%C3%A8me_d%27exploitation))

Ubuntu (prononciation : /u.bun.tu/) est un système d'exploitation GNU/Linux basé sur la distribution Linux Debian. Il est développé, commercialisé et maintenu pour les ordinateurs individuels par la société Canonical.

Ubuntu se définit comme « un système d'exploitation utilisé par des millions de PC à travers le monde »¹⁰ et avec une interface « simple, intuitive, et sécurisée ».

Elle est la distribution la plus consultée sur Internet d'après le site Alexa. Et est le système d'exploitation le plus utilisé sur les systèmes Cloud ainsi que sur les serveurs informatiques.

Ubuntu se divise en deux branches :

- La branche principale stable dit LTS. Avec mise à niveau tous les six mois et mise à jour majeure tous les 2 ans. La dernière version 16.04.3, nom de code *Xenial Xerus* est sortie le 3 août 2017.
- La branche secondaire instable avec mise à jour majeure tous les six mois. La dernière version en date est la 17.10, nom de code *Artful Aardvark** est sortie le 19 octobre 2017.

23.4.3.3 La Philosophie d'Ubuntu

Le mot *ubuntu* provient d'un ancien mot bantou (famille de langues africaines) qui désigne une personne qui prend conscience que son *moi* est intimement lié à ce que sont les autres. Autrement dit : **Je suis ce que je suis grâce à ce que nous sommes tous.**

C'est un concept fondamental de la « philosophie de la réconciliation » développée par Desmond Mpilo Tutu avec l'abolition de l'apartheid.

Ubuntu signifie par ailleurs en kinyarwanda (langue rwandaise) et en kirundi (langue burundaise) *humanité, générosité ou gratuité*.

On dit d'une chose qu'elle est *k'ubuntu* si elle est obtenue gratuitement.

En informatique, on considère qu'une distribution existe aux travers des apports des différentes communautés Linux. Et tel qu'il se trouve expliqué dans le travail de la Commission de la vérité et de la réconciliation. Elles permettent de mieux saisir par exemple la mission de la Fondation Shuttleworth relayée en France par les travaux de philosophes comme Barbara Cassin et Philippe-Joseph Salazar.

23.4.4 Images CentOS

See also:

- <https://store.docker.com/images/centos>
- https://hub.docker.com/_/centos/
- <https://fr.wikipedia.org/wiki/CentOS>
- <https://www.centos.org/>

Contents

- *Images CentOS*
 - *Short Description*
 - *Description*
 - *Structures*

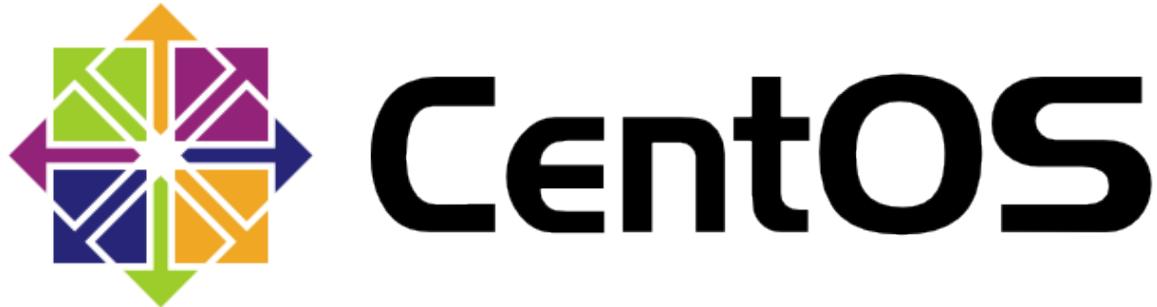


Fig. 9: Logo CentOS

A screenshot of a web browser showing the Docker Hub page for the "centos" repository. The URL in the address bar is https://hub.docker.com/_/centos/. The page title is "OFFICIAL REPOSITORY". Below the title, the repository name "centos" is displayed with a star icon. A note says "Last pushed: 7 days ago". There are two tabs at the top: "Repo Info" (which is active) and "Tags".

Under "Repo Info", there are two sections: "Short Description" which contains "The official build of CentOS;" and "Docker Pull Command" which contains "docker pull centos".

Under "Tags", there is a section titled "Supported tags and respective Dockerfile links" containing a bulleted list of tags:

- latest , centos7 , 7 ([docker/Dockerfile](#))
- centos6 , 6 ([docker/Dockerfile](#))
- centos7.4.1708 , 7.4.1708 ([docker/Dockerfile](#))
- centos7.3.1611 , 7.3.1611 ([docker/Dockerfile](#))
- centos7.2.1511 , 7.2.1511 ([docker/Dockerfile](#))
- centos7.1.1503 , 7.1.1503 ([docker/Dockerfile](#))
- centos7.0.1406 , 7.0.1406 ([docker/Dockerfile](#))
- centos6.9 , 6.9 ([docker/Dockerfile](#))
- centos6.8 , 6.8 ([docker/Dockerfile](#))
- centos6.7 , 6.7 ([docker/Dockerfile](#))
- centos6.6 , 6.6 ([docker/Dockerfile](#))

Fig. 10: https://hub.docker.com/_/centos/

23.4.4.1 Short Description

The official build of CentOS.

23.4.4.2 Description

See also:

- <https://fr.wikipedia.org/wiki/CentOS>

CentOS (Community enterprise Operating System) est une distribution GNU/Linux principalement destinée aux serveurs. Tous ses paquets, à l'exception du logo, sont des paquets compilés à partir des sources de la distribution RHEL (Red Hat Enterprise Linux), éditée par la société Red Hat. Elle est donc quasiment identique à celle-ci et se veut 100 % compatible d'un point de vue binaire.

Utilisée par 20 % des serveurs web Linux, elle est l'une des distributions Linux les plus populaires pour les serveurs web. Depuis novembre 2013, elle est la troisième distribution la plus utilisée sur les serveurs web ; en avril 2017, elle était installée sur 20,6 % d'entre eux ; les principales autres distributions étaient Debian (31,8 %), Ubuntu (35,8 %) et Red Hat (3,3 %).

23.4.4.3 Structures

La RHEL en version binaire, directement installable et exploitable, ne peut être obtenue que par achat d'une souscription auprès de Red Hat ou de ses revendeurs. La plupart des programmes inclus et livrés avec la Red Hat sont publiés sous la licence GPL, qui impose au redistributeur (sous certaines conditions) de fournir les sources. CentOS utilise donc les sources de la RHEL (accessibles librement sur Internet) pour regénérer la Red Hat à l'identique.

On peut donc considérer la CentOS comme une version gratuite de la Red Hat. Le support technique est de type communautaire : il se fait gratuitement et ouvertement via les listes de diffusion et les forums de la communauté CentOS.

Depuis le 7 janvier 2014, Red Hat et CentOS se sont fortement rapprochées, puisque la plupart des principaux membres maintenant la CentOS ont été embauchés par Red Hat.

23.5 Images langages

23.5.1 Images Python

See also:

- <https://store.docker.com/images/python>
- https://hub.docker.com/_/python/

Contents

- *Images Python*
 - *Short Description*
 - *What is Python ?*
 - *How to use this image*



Fig. 11: Le logo Python

A screenshot of the Docker Hub website showing the details for the "python" image. The page includes the Python logo, the name "python" in bold, the maintainer "By Docker", a description stating "Python is an interpreted, interactive, object-oriented, open-source programming language.", a badge indicating "10M+" downloads, and filters for "Container", "Linux", "x86-64", "IBM Z", "IBM POWER", and "Programming Languages". On the right, there's information about the "Official Image" being free (\$0.00), a "Terms of Service" link, a dropdown for "Linux - x86-64", a "Copy and paste to pull this image" button containing the command "docker pull python", and a "View Available Tags" link. Below this, there's a "DESCRIPTION" section with tabs for "DESCRIPTION", "REVIEWS", and "RESOURCES". The "DESCRIPTION" tab shows a list of "Supported tags and respective Dockerfile links" with entries like "3.7.0a4-stretch", "3.7-rc-stretch", "rc-stretch (3.7-rc/stretch/Dockerfile)", and "3.7.0a4-slim-stretch", etc. To the right of this is a "Reviews" section showing an average rating of 5 stars from 4 ratings, a "Add Product Review" form, and a "Start Your Review" button.

Fig. 12: <https://store.docker.com/images/python>

23.5.1.1 Short Description

Python is an interpreted, interactive, object-oriented, open-source programming language.

23.5.1.2 What is Python ?

Python is an interpreted, interactive, object-oriented, open-source programming language.

It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Python combines remarkable power with very clear syntax.

It has interfaces to many system calls and libraries, as well as to various window systems, and is extensible in C or C++.

It is also usable as an extension language for applications that need a programmable interface.

Finally, Python is portable: it runs on many Unix variants, on the Mac, and on Windows 2000 and later.

23.5.1.3 How to use this image

Create a Dockerfile in your Python app project

```
FROM python:3

WORKDIR /usr/src/app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD [ "python", "./your-daemon-or-script.py" ]
```

You can then build **and** run the Docker image:

```
docker build -t my-python-app .
```

```
docker run -it --rm --name my-running-app my-python-app
```

Run a single Python script

For many simple, single file projects, you may find it inconvenient to write a complete Dockerfile. In such cases, you can run a Python script by using the Python Docker image directly:

```
docker run -it --rm --name my-running-script -v "$PWD":/usr/src/myapp -w /usr/src/
˓→myapp python:3 python your-daemon-OR-script.py
```

23.5.2 Images pipenv

See also:

- <https://hub.docker.com/r/kennethreitz/pipenv/>

Contents

- *Images pipenv*
 - *Short Description*
 - *What is Python ?*
 - *Dockerfile*
 - *How to use this image*

23.5.2.1 Short Description

Pipenv Docker image.

23.5.2.2 What is Python ?

Pipenv is a tool that aims to bring the best of all packaging worlds (bundler, composer, npm, cargo, yarn, etc.) to the Python world. Windows is a first-class citizen, in our world.

It automatically creates and manages a virtualenv for your projects, as well as adds/removes packages from your Pipfile as you install/uninstall packages. It also generates the ever-important Pipfile.lock, which is used to produce deterministic builds.

23.5.2.3 Dockerfile

See also:

- <https://github.com/pypa/pipenv/blob/master/Dockerfile>

```
FROM heroku/heroku:18-build

ENV DEBIAN_FRONTEND noninteractive
ENV LC_ALL C.UTF-8
ENV LANG C.UTF-8

# -- Install Pipenv:
RUN apt update && apt upgrade -y && apt install python3.7-dev -y
RUN curl --silent https://bootstrap.pypa.io/get-pip.py | python3.7

# Backwards compatibility.
RUN rm -fr /usr/bin/python3 && ln /usr/bin/python3.7 /usr/bin/python3

RUN pip3 install pipenv

# -- Install Application into container:
RUN set -ex && mkdir /app

WORKDIR /app

# -- Adding Pipfiles
ONBUILD COPY Pipfile Pipfile
ONBUILD COPY Pipfile.lock Pipfile.lock
```

(continues on next page)

(continued from previous page)

```
# -- Install dependencies:  
ONBUILD RUN set -ex && pipenv install --deploy --system  
  
# -----  
# - Using This File: -  
# -----  
  
# FROM kennethreitz/pipenv  
  
# COPY . /app  
  
# -- Replace with the correct path to your app's main executable  
# CMD python3 main.py
```

23.5.2.4 How to use this image

See also:

- <http://python-responder.org/en/latest/deployment.html#docker-deployment>

```
from kennethreitz/pipenv  
  
COPY . /app  
CMD python3 api.py
```

23.5.3 Images PHP

See also:

- <https://store.docker.com/images/php>
- <https://en.wikipedia.org/wiki/PHP>

Contents

- *Images PHP*
 - *Short Description*
 - *What is PHP ?*



Fig. 13: Le logo PHP

23.5.3.1 Short Description

While designed for web development, the PHP scripting language also provides general-purpose use.

23.5.3.2 What is PHP ?

See also:

- <https://en.wikipedia.org/wiki/PHP>

PHP is a server-side scripting language designed for web development, but which can also be used as a general-purpose programming language. PHP can be added to straight HTML or it can be used with a variety of templating engines and web frameworks.

PHP code is usually processed by an interpreter, which is either implemented as a native module on the web-server or as a common gateway interface (CGI).

23.5.4 Images Ruby

See also:

- <https://store.docker.com/images/ruby>
- https://en.wikipedia.org/wiki/Ruby_%28programming_language%29

Contents

- *Images Ruby*
 - *Short Description*
 - *What is Ruby ?*



Fig. 14: Le logo Ruby

23.5.4.1 Short Description

Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.

23.5.4.2 What is Ruby ?

See also:

- https://en.wikipedia.org/wiki/Ruby_%28programming_language%29

Ruby is a dynamic, reflective, object-oriented, general-purpose, open-source programming language.

According to its authors, Ruby was influenced by Perl, Smalltalk, Eiffel, Ada, and Lisp. It supports multiple programming paradigms, including functional, object-oriented, and imperative. It also has a dynamic type system and automatic memory management.

23.5.5 Images Node

See also:

- <https://store.docker.com/images/node>
- https://en.wikipedia.org/wiki/Ruby_%28programming_language%29

Contents

- *Images Node*
 - *Short Description*
 - *What is Node.js ?*



Fig. 15: Le logo Node.js

23.5.5.1 Short Description

Node.js is a JavaScript-based platform for server-side and networking applications.

23.5.5.2 What is Node.js ?

See also:

- <https://en.wikipedia.org/wiki/Node.js>

Node.js is a software platform for scalable server-side and networking applications.

Node.js applications are written in JavaScript and can be run within the Node.js runtime on Mac OS X, Windows, and Linux without changes.

Node.js applications are designed to maximize throughput and efficiency, using non-blocking I/O and asynchronous events.

Node.js applications run single-threaded, although Node.js uses multiple threads for file and network events.

Node.js is commonly used for real-time applications due to its asynchronous nature.

Node.js internally uses the Google V8 JavaScript engine to execute code; a large percentage of the basic modules are written in JavaScript.

Node.js contains a built-in, asynchronous I/O library for file, socket, and HTTP communication.

The HTTP and socket support allows Node.js to act as a **web server** without additional software such as Apache.

23.5.6 Images Go (Golang)

See also:

- <https://store.docker.com/images/golang>
- https://en.wikipedia.org/wiki/Go_%28programming_language%29

Contents

- *Images Go (Golang)*
 - *Short Description*
 - *What is Go ?*

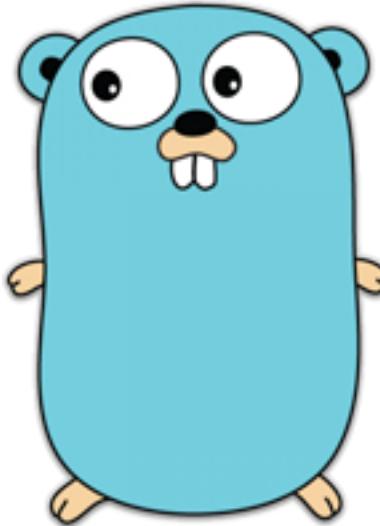


Fig. 16: Le logo Golang

23.5.6.1 Short Description

Node.js is a JavaScript-based platform for server-side and networking applications.

23.5.6.2 What is Go ?

See also:

- https://en.wikipedia.org/wiki/Go_%28programming_language%29

Go (a.k.a., Golang) is a programming language first developed at Google.

It is a statically-typed language with syntax loosely derived from C, but with additional features such as garbage collection, type safety, some dynamic-typing capabilities, additional built-in types (e.g., variable-length arrays and key-value maps), and a large standard library.

23.5.7 Images OpenJDK (Java)

See also:

- https://hub.docker.com/_/openjdk/
- <https://en.wikipedia.org/wiki/OpenJDK>
- <http://openjdk.java.net/>
- <https://github.com/docker-library/openjdk>
- <https://twitter.com/OpenJDK>

Contents

- *Images OpenJDK (Java)*
 - *Short Description*
 - *What is OpenJDK ?*
 - *How to use this image*



Fig. 17: Le logo OpenJDK

23.5.7.1 Short Description

OpenJDK is an open-source implementation of the Java Platform, Standard Edition

23.5.7.2 What is OpenJDK ?

See also:

- <https://en.wikipedia.org/wiki/OpenJDK>

OpenJDK (Open Java Development Kit) is a free and open source implementation of the Java Platform, Standard Edition (Java SE).

OpenJDK is the official reference implementation of Java SE since version 7.

23.5.7.3 How to use this image

Start a Java instance in your app

The most straightforward way to use this image is to use a Java container as both the build and runtime environment. In your Dockerfile, writing something along the lines of the following will compile and run your project:

Full Description

Supported tags and respective [Dockerfile links](#)

Simple Tags

- 10-ea-32-jre-experimental, 10-ea-jre-experimental, 10-jre-experimental, 10-ea-32-jre, 10-ea-jre, 10-jre ([10-jre/Dockerfile](#))
- 10-ea-32-jre-slim-experimental, 10-ea-jre-slim-experimental, 10-jre-slim-experimental, 10-ea-32-jre-slim, 10-ea-jre-slim, 10-jre-slim ([10-jre/slim/Dockerfile](#))
- 10-ea-32-jdk-experimental, 10-ea-32-experimental, 10-ea-jdk-experimental, 10-ea-experimental, 10-jdk-experimental, 10-experimental, 10-ea-32-jdk, 10-ea-32, 10-ea-jdk, 10-ea, 10-jdk, 10 ([10-jdk/Dockerfile](#))
- 10-ea-32-jdk-slim-experimental, 10-ea-32-slim-experimental, 10-ea-jdk-slim-experimental, 10-ea-slim-experimental, 10-jdk-slim-experimental, 10-slim-experimental, 10-ea-32-jdk-slim, 10-ea-32-slim, 10-ea-jdk-slim, 10-ea-slim, 10-jdk-slim, 10-slim ([10-jdk/slim/Dockerfile](#))
- 9.0.1-11-jre-sid, 9.0.1-jre-sid, 9.0-jre-sid, 9-jre-sid, 9.0.1-11-jre, 9.0.1-jre, 9.0-jre, 9-jre ([9-jre/Dockerfile](#))
- 9.0.1-11-jre-slim-sid, 9.0.1-jre-slim-sid, 9.0-jre-slim-sid, 9-jre-slim-sid, 9.0.1-11-jre-slim, 9.0.1-jre-slim, 9.0-jre-slim, 9-jre-slim ([9-jre/slim/Dockerfile](#))
- 9.0.1-11-jdk-sid, 9.0.1-11-sid, 9.0.1-jdk-sid, 9.0.1-sid, 9.0-jdk-sid, 9.0-sid, 9-jdk-sid, 9-sid, 9.0.1-11-jdk, 9.0.1-11, 9.0.1-jdk, 9.0.1-jre, 9.0.1-slim, 9.0.1-jre-slim, 9.0.1-jdk-slim, 9.0.1-slim ([9-jdk/Dockerfile](#))
- 9.0.1-11-jdk-slim-sid, 9.0.1-11-slim-sid, 9.0.1-jdk-slim-sid, 9.0.1-slim-sid, 9.0.1-11-jdk-slim, 9.0.1-11-slim, 9.0.1-jdk-slim, 9.0.1-slim, 9-jdk-slim, 9-slim ([9-jdk/slim/Dockerfile](#))
- 9.0.1-jdk-windowsservercore-ltsc2016, 9.0.1-windowsservercore-ltsc2016, 9.0-jdk-windowsservercore-ltsc2016, 9.0-windowsservercore-ltsc2016, 9-jdk-windowsservercore-

Fig. 18: https://hub.docker.com/_/openjdk/

```
FROM openjdk:7
COPY . /usr/src/myapp
WORKDIR /usr/src/myapp
RUN javac Main.java
CMD ["java", "Main"]
```

You can then run and build the Docker image:

```
$ docker build -t my-java-app .
$ docker run -it --rm --name my-running-app my-java-app
```

23.6 Images webserver: serveurs HTTP (serveurs Web)

See also:

- https://fr.wikipedia.org/wiki/Serveur_HTTP
- https://fr.wikipedia.org/wiki/Serveur_HTTP#Logiciels_de_serveur_HTTP
- https://en.wikipedia.org/wiki/Web_server

Le serveur HTTP le plus utilisé est Apache HTTP Server qui sert environ 55% des sites web en janvier 2013 selon Neteckraft.

Le serveur HTTP le plus utilisé dans les 1 000 sites les plus actifs est en revanche Nginx avec 38,2% de parts de marché en 2016 selon w3techs et 53,9% en avril 2017.

23.6.1 Images Apache HTTPD

See also:

- https://hub.docker.com/_/httpd/
- https://en.wikipedia.org/wiki/Apache_HTTP_Server
- <https://httpd.apache.org/>

Contents

- *Images Apache HTTPD*
 - *Short Description*
 - *What is httpd ?*
 - *Configuration*
 - *SSL/HTTPS*

23.6.1.1 Short Description

The Apache HTTP Server Project.



Fig. 19: Le logo Apache HTTPD

Full Description

Supported tags and respective [Dockerfile](#) links

- 2.2.34 , 2.2 ([2.2/Dockerfile](#))
- 2.2.34-alpine , 2.2-alpine ([2.2/alpine/Dockerfile](#))
- 2.4.29 , 2.4 , 2 , latest ([2.4/Dockerfile](#))
- 2.4.29-alpine , 2.4-alpine , 2-alpine , alpine ([2.4/alpine/Dockerfile](#))

Fig. 20: https://hub.docker.com/_/httpd/

23.6.1.2 What is httpd ?

The Apache HTTP Server, colloquially called **Apache**, is a Web server application notable for playing a key role in the initial growth of the World Wide Web.

Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled.

Apache quickly overtook NCSA HTTPd as the dominant HTTP server, and has remained the most popular HTTP server in use since April 1996.

23.6.1.3 Configuration

To customize the configuration of the httpd server, just COPY your custom configuration in as /usr/local/apache2/conf/httpd.conf.

```
FROM httpd:2.4
COPY ./my-httdp.conf /usr/local/apache2/conf/httpd.conf
```

23.6.1.4 SSL/HTTPS

If you want to run your web traffic over SSL, the simplest setup is to COPY or mount (-v) your server.crt and server.key into /usr/local/apache2/conf/ and then customize the /usr/local/apache2/conf/httpd.conf by removing the comment symbol from the following lines:

```
...
#LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
...
#LoadModule ssl_module modules/mod_ssl.so
...
#Include conf/extra/httpd-ssl.conf
...
```

The conf/extra/httpd-ssl.conf configuration file will use the certificate files previously added and tell the daemon to also listen on port 443.

Be sure to also add something like -p 443:443 to your docker run to forward the https port.

This could be accomplished with a sed line similar to the following:

```
RUN sed -i \
    -e 's/^#\ \(Include .*httpd-ssl.conf\)/\1/' \
    -e 's/^#\ \(LoadModule .*mod_ssl.so\)/\1/' \
    -e 's/^#\ \(LoadModule .*mod_socache_shmcb.so\)/\1/' \
    conf/httpd.conf
```

The previous steps should work well for development, but we recommend customizing your conf files for production, see httpd.apache.org for more information about SSL setup.

23.6.2 Images Apache HTTPD bitnami

See also:

- <https://twitter.com/Bitnami>
- <https://hub.docker.com/r/bitnami/apache/>

- <https://github.com/bitnami/bitnami-docker-apache>
- https://en.wikipedia.org/wiki/Apache_HTTP_Server
- <https://httpd.apache.org/>

Contents

- *Images Apache HTTPD bitnami*
 - *Short Description*
 - * *What is Apache ?*
 - *TL;DR;*
 - *Docker Compose*
 - *Dockerfile*
 - *Why use Bitnami Images ?*
 - *Adding custom virtual hosts*
 - * *Step 1: Write your my_vhost.conf file with the following content*
 - * *Step 2: Mount the configuration as a volume*
 - *Using custom SSL certificates*
 - * *Step 1: Prepare your certificate files*
 - * *Step 2: Run the Apache image*
 - *Full configuration*
 - * *Step 1: Run the Apache image*
 - * *Step 2: Edit the configuration*
 - * *Step 3: Restart Apache*
 - *Logging*
 - *Upgrade this image*
 - * *Step 1: Get the updated image*
 - * *Step 2: Stop and backup the currently running container*
 - * *Step 3: Remove the currently running container*
 - * *Step 4: Run the new image*
 - *Notable Changes*
 - * *2.4.34-r8 (2018-07-24)*
 - * *2.4.18-r0*
 - * *2.4.12-4-r01*

23.6.2.1 Short Description

23.6.2.1.1 What is Apache ?

The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT.

The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

23.6.2.2 TL;DR;

```
$ docker run --name apache bitnami/apache:latest
```

23.6.2.3 Docker Compose

```
$ curl -sSL https://raw.githubusercontent.com/bitnami/bitnami-docker-apache/master/  
→docker-compose.yml > docker-compose.yml
```

```
$ docker-compose up -d
```

23.6.2.4 Dockerfile

```
FROM bitnami/minideb-extras:stretch-r158  
LABEL maintainer "Bitnami <containers@bitnami.com>"  
  
ENV BITNAMI_PKG_CHMOD="-R g+rwx" \  
    BITNAMI_PKG_EXTRA_DIRS="/bitnami/apache/conf /opt/bitnami/apache/tmp /opt/bitnami/  
→apache/conf" \  
    HOME="/" \  
  
# Install required system packages and dependencies  
RUN install_packages libc6 libexpat1 libffi6 libgmp10 libgnutls30 libhogweed4_  
→libidn11 libldap-2.4-2 libnettle6 libp11-kit0 libpcre3 libsasl2-2 libssl1.1_  
→libtasn1-6 zlib1g  
RUN bitnami-pkg unpack apache-2.4.35-0 --checksum_  
→1e352e2185137fcad60bb6fdf2961368f59a35e2c2cab4ee94c77152f1c37299  
RUN ln -sf /opt/bitnami/apache/htdocs /app  
RUN ln -sf /dev/stdout /opt/bitnami/apache/logs/access_log  
RUN ln -sf /dev/stdout /opt/bitnami/apache/logs/error_log  
  
COPY rootfs /  
ENV APACHE_HTTPS_PORT_NUMBER="8443" \  
    APACHE_HTTP_PORT_NUMBER="8080" \  
    BITNAMI_APP_NAME="apache" \  
    BITNAMI_IMAGE_VERSION="2.4.35-debian-9-r10" \  
    PATH="/opt/bitnami/apache/bin:$PATH"  
  
EXPOSE 8080 8443  
  
WORKDIR /app  
USER 1001
```

(continues on next page)

(continued from previous page)

```
ENTRYPOINT [ "/app-entrypoint.sh" ]
CMD [ "/run.sh" ]
```

23.6.2.5 Why use Bitnami Images ?

Bitnami closely tracks upstream source changes and promptly publishes new versions of this image using our automated systems.

With Bitnami images the latest bug fixes and features are available as soon as possible.

Bitnami containers, virtual machines and cloud images use the same components and configuration approach - making it easy to switch between formats based on your project needs.

Bitnami images are built on CircleCI and automatically pushed to the Docker Hub.

All our images are based on minideb a minimalist Debian based container image which gives you a small base container image and the familiarity of a leading linux distribution.

23.6.2.6 Adding custom virtual hosts

The default httpd.conf includes virtual hosts placed in /bitnami/apache/conf/vhosts/.

You can mount a my_vhost.conf file containing your custom virtual hosts at this location.

23.6.2.6.1 Step 1: Write your my_vhost.conf file with the following content

```
<VirtualHost *:8080>
  ServerName www.example.com
  DocumentRoot "/app"
  <Directory "/app">
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
  </Directory>
</VirtualHost>
```

23.6.2.6.2 Step 2: Mount the configuration as a volume

```
$ docker run --name apache \
-v /path/to/my_vhost.conf:/bitnami/apache/conf/vhosts/my_vhost.conf:ro \
bitnami/apache:latest
```

or using Docker Compose:

```
version: '2'

services:
  apache:
    image: 'bitnami/apache:latest'
    ports:
      - '80:8080'
```

(continues on next page)

(continued from previous page)

```
- '443:8443'  
volumes:  
- /path/to/my_vhost.conf:/bitnami/apache/conf/vhosts/my_vhost.conf:ro
```

23.6.2.7 Using custom SSL certificates

Note: The steps below assume that you are using a custom domain name and that you have already configured the custom domain name to point to your server.

This container comes with SSL support already pre-configured and with a dummy certificate in place (server.crt and server.key files in /bitnami/apache/conf/bitnami/certs).

If you want to use your own certificate (.crt) and certificate key (.key) files, follow the steps below:

23.6.2.7.1 Step 1: Prepare your certificate files

In your local computer, create a folder called certs and put your certificates files. Make sure you rename both files to server.crt and server.key respectively:

```
$ mkdir /path/to/apache-persistence/apache/conf/bitnami/certs -p  
$ cp /path/to/certfile.crt /path/to/apache-persistence/apache/conf/bitnami/certs  
cp /path/to/keyfile.key /path/to/apache-persistence/apache/conf/bitnami/certs/server.  
→key
```

23.6.2.7.2 Step 2: Run the Apache image

Run the Apache image, mounting the certificates directory from your host.

```
$ docker run --name apache \  
-v /path/to/apache-persistence/apache/conf/bitnami/certs:/bitnami/apache/conf/  
→bitnami/certs \  
bitnami/apache:latest
```

or using Docker Compose:

```
version: '2'  
  
services:  
  apache:  
    image: 'bitnami/apache:latest'  
    ports:  
      - '80:8080'  
      - '443:8443'  
    volumes:  
      - /path/to/apache-persistence/apache/conf/bitnami/certs:/bitnami/apache/conf/  
→bitnami/certs
```

23.6.2.8 Full configuration

The image looks for configurations in /bitnami/apache/conf/.

You can mount a volume at /bitnami and copy/edit the configurations in the /bitnami/apache/conf/. The default configurations will be populated in the conf/ directory if it's empty.

23.6.2.8.1 Step 1: Run the Apache image

Run the Apache image, mounting a directory from your host.

```
$ docker run --name apache \
-v /path/to/apache-persistence:/bitnami \
bitnami/apache:latest
```

or using Docker Compose:

```
version: '2'

services:
  apache:
    image: 'bitnami/apache:latest'
    ports:
      - '80:8080'
      - '443:8443'
    volumes:
      - /path/to/apache-persistence:/bitnami
```

23.6.2.8.2 Step 2: Edit the configuration

Edit the configuration on your host using your favorite editor.

```
$ vi /path/to/apache-persistence/apache/conf/httpd.conf
```

23.6.2.8.3 Step 3: Restart Apache

After changing the configuration, restart your Apache container for the changes to take effect.

```
$ docker restart apache
```

or using Docker Compose:

```
$ docker-compose restart apache
```

23.6.2.9 Logging

The Bitnami Apache Docker image sends the container logs to the stdout. To view the logs:

```
$ docker logs apache
```

or using Docker Compose:

```
$ docker-compose logs apache
```

You can configure the containers logging driver using the `--log-driver` option if you wish to consume the container logs differently.

In the default configuration docker uses the json-file driver.

23.6.2.10 Upgrade this image

Bitnami provides up-to-date versions of Apache, including security patches, soon after they are made upstream. We recommend that you follow these steps to upgrade your container.

23.6.2.10.1 Step 1: Get the updated image

```
$ docker pull bitnami/apache:latest
```

or if you're using Docker Compose, update the value of the image property to `bitnami/apache:latest`.

23.6.2.10.2 Step 2: Stop and backup the currently running container

Stop the currently running container using the command

```
$ docker stop apache
```

or using Docker Compose:

```
$ docker-compose stop apache
```

Next, take a snapshot of the persistent volume `/path/to/apache-persistence` using:

```
$ rsync -a /path/to/apache-persistence /path/to/apache-persistence.bkp.$(date +%Y%m%d-%H.%M.%S)
```

You can use this snapshot to restore the database state should the upgrade fail ??.

23.6.2.10.3 Step 3: Remove the currently running container

```
$ docker rm -v apache
```

or using Docker Compose:

```
$ docker-compose rm -v apache
```

23.6.2.10.4 Step 4: Run the new image

Re-create your container from the new image.

```
$ docker run --name apache bitnami/apache:latest
```

or using Docker Compose:

```
$ docker-compose up apache
```

23.6.2.11 Notable Changes

23.6.2.11.1 2.4.34-r8 (2018-07-24)

See also:

- <https://github.com/bitnami/bitnami-docker-apache/tree/2.4.34-ol-7-r8>

The Apache container has been migrated to a non-root user approach.

Previously the container ran as the root user and the Apache daemon was started as the apache user.

From now on, both the container and the Apache daemon run as user 1001.

As a consequence, the HTTP/HTTPS ports exposed by the container are now **8080/8443 instead of 80/443**

You can revert this behavior by changing USER 1001 to USER root in the Dockerfile.

23.6.2.11.2 2.4.18-r0

The configuration volume has been moved to /bitnami/apache.

Now you only need to mount a single volume at /bitnami/apache for persisting configuration.

/app is still used for serving content by the default virtual host.

The logs are always sent to the stdout and are no longer collected in the volume.

23.6.2.11.3 2.4.12-4-r01

The /app directory is no longer exported as a volume.

This caused problems when building on top of the image, since changes in the volume are not persisted between Dockerfile RUN instructions.

To keep the previous behavior (so that you can mount the volume in another container), create the container with the -v /app option.

23.6.3 Images apache Tomcat

See also:

- https://hub.docker.com/_/tomcat/
- https://fr.wikipedia.org/wiki/Apache_Tomcat

Contents

- *Images apache Tomcat*
 - *Short Description*
 - *What is Apache Tomcat ?*

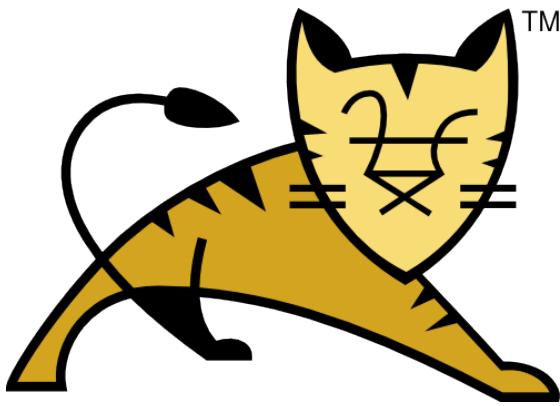


Fig. 21: Le logo Apache Tomcat

23.6.3.1 Short Description

Apache Tomcat is an open source implementation of the Java Servlet and JavaServer Pages technologies

23.6.3.2 What is Apache Tomcat ?

Apache Tomcat (or simply Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF).

Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Oracle, and provides a “pure Java” HTTP web server environment for Java code to run in.

In the simplest config Tomcat runs in a single operating system process.

The process runs a Java virtual machine (JVM).

Every single HTTP request from a browser to Tomcat is processed in the Tomcat process in a separate thread.

23.6.4 Images webserver : serveurs Web + reverse proxy + load balancer

See also:

- https://en.wikipedia.org/wiki/Reverse_proxy
- https://fr.wikipedia.org/wiki/Proxy_inverse
- https://en.wikipedia.org/wiki/Load_balancer
- https://en.wikipedia.org/wiki/HTTP_cache

See also:

- https://en.wikipedia.org/wiki/Web_server

23.6.4.1 Apache HTTP Server + mod_proxy

Apache HTTP Server, serveur HTTP libre configurable en *proxy inverse* avec le module mod_proxy.

Supported tags and respective Dockerfile links

- 7.0.82-jre7, 7.0-jre7, 7-jre7, 7.0.82, 7.0, 7 ([7/jre7/Dockerfile](#))
- 7.0.82-jre7-slim, 7.0-jre7-slim, 7-jre7-slim, 7.0.82-slim, 7.0-slim, 7-slim ([7/jre7-slim/Dockerfile](#))
- 7.0.82-jre7-alpine, 7.0-jre7-alpine, 7-jre7-alpine, 7.0.82-alpine, 7.0-alpine, 7-alpine ([7/jre7-alpine/Dockerfile](#))
- 7.0.82-jre8, 7.0-jre8, 7-jre8 ([7/jre8/Dockerfile](#))
- 7.0.82-jre8-slim, 7.0-jre8-slim, 7-jre8-slim ([7/jre8-slim/Dockerfile](#))
- 7.0.82-jre8-alpine, 7.0-jre8-alpine, 7-jre8-alpine ([7/jre8-alpine/Dockerfile](#))
- 8.0.48-jre7, 8.0-jre7, 8.0.48, 8.0 ([8.0/jre7/Dockerfile](#))
- 8.0.48-jre7-slim, 8.0-jre7-slim, 8.0.48-slim, 8.0-slim ([8.0/jre7-slim/Dockerfile](#))
- 8.0.48-jre7-alpine, 8.0-jre7-alpine, 8.0.48-alpine, 8.0-alpine ([8.0/jre7-alpine/Dockerfile](#))
- 8.0.48-jre8, 8.0-jre8 ([8.0/jre8/Dockerfile](#))
- 8.0.48-jre8-slim, 8.0-jre8-slim ([8.0/jre8-slim/Dockerfile](#))
- 8.0.48-jre8-alpine, 8.0-jre8-alpine ([8.0/jre8-alpine/Dockerfile](#))
- 8.5.24-jre8, 8.5-jre8, 8-jre8, jre8, 8.5.24, 8.5, 8, latest ([8.5/jre8/Dockerfile](#))
- 8.5.24-jre8-slim, 8.5-jre8-slim, 8-jre8-slim, jre8-slim, 8.5.24-slim, 8.5-slim, 8-slim, slim ([8.5/jre8-slim/Dockerfile](#))
- 8.5.24-jre8-alpine, 8.5-jre8-alpine, 8-jre8-alpine, jre8-alpine, 8.5.24-alpine, 8.5-alpine, 8-alpine, alpine ([8.5/jre8-alpine/Dockerfile](#))
- 8.5.24-jre9, 8.5-jre9, 8-jre9, jre9 ([8.5/jre9/Dockerfile](#))
- 8.5.24-jre9-slim, 8.5-jre9-slim, 8-jre9-slim, jre9-slim ([8.5/jre9-slim/Dockerfile](#))
- 9.0.2-jre8, 9.0-jre8, 9-jre8 ([9.0/jre8/Dockerfile](#))
- 9.0.2-jre8-slim, 9.0-jre8-slim, 9-jre8-slim ([9.0/jre8-slim/Dockerfile](#))
- 9.0.2-jre8-alpine, 9.0-jre8-alpine, 9-jre8-alpine ([9.0/jre8-alpine/Dockerfile](#))
- 9.0.2-jre9, 9.0-jre9, 9-jre9, 9.0.2, 9.0, 9 ([9.0/jre9/Dockerfile](#))
- 9.0.2-jre9-slim, 9.0-jre9-slim, 9-jre9-slim, 9.0.2-slim, 9.0-slim, 9-slim ([9.0/jre9-slim/Dockerfile](#))

Fig. 22: https://hub.docker.com/_/tomcat/

23.6.4.2 Nginx

23.6.4.2.1 Images nginx (engine-x)

See also:

- https://hub.docker.com/_/nginx/
- <https://en.wikipedia.org/wiki/Nginx>

Contents

- *Images nginx (engine-x)*
 - *Short Description*
 - *What is nginx ?*



Fig. 23: Le logo Nginx

Short Description

Official build of Nginx.

What is nginx ?

See also:

- https://en.wikipedia.org/wiki/Web_server
- https://en.wikipedia.org/wiki/Reverse_proxy
- https://en.wikipedia.org/wiki/Load_balancer
- https://en.wikipedia.org/wiki/HTTP_cache

Nginx (pronounced “engine-x”) is an:

- open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols,
- as well as a load balancer, HTTP cache,
- and a **web server** (origin server).

The nginx project started with a strong focus on high concurrency, high performance and low memory usage.

It is licensed under the 2-clause BSD-like license and it runs on Linux, BSD variants, Mac OS X, Solaris, AIX, HP-UX, as well as on other nix flavors.

It also has a proof of concept port for Microsoft Windows.

A large fraction of web servers use NGINX often as a load balancer.

23.7 Images db : bases de données

23.7.1 Images PostgreSQL

See also:

- <https://store.docker.com/images/postgres>
- https://hub.docker.com/_/postgres/
- <https://fr.wikipedia.org/wiki/PostgreSQL>
- <https://en.wikipedia.org/wiki/PostgreSQL>

Contents

- *Images PostgreSQL*
 - *Short Description*
 - *Description*
 - *What is PostgreSQL ?*
 - *Environment Variables*
 - * *POSTGRES_PASSWORD*
 - * *POSTGRES_USER*
 - * *PGDATA*
 - * *POSTGRES_DB*
 - * *POSTGRES_INITDB_WALDIR*
 - *Docker Secrets*
 - *How to extend this image*
 - * *Extends with a Dockerfile*
 - *docker-compose up*

23.7.1.1 Short Description

The PostgreSQL object-relational database system provides reliability and data integrity.

23.7.1.2 Description

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD.

Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB, MySQL et Firebird), ou propriétaires (comme Oracle, Sybase, DB2, Informix et Microsoft SQL Server).

Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises



Fig. 24: Le logo PostgreSQL

The screenshot shows the Docker Hub page for the 'postgres' image. At the top, it says 'Official Image \$0.00'. Below that is a 'Terms of Service' link and a dropdown menu set to 'Linux - x86-64'. A button labeled 'Copy and paste to pull this image' contains the command 'docker pull postgres'. Below this is a 'View Available Tags' link. The main content area has tabs for 'DESCRIPTION', 'REVIEWS', and 'RESOURCES'. The 'DESCRIPTION' tab is active and displays a section titled 'Supported tags and respective Dockerfile links' with a list of tags and their Dockerfile links, such as '10.1', '10.1-alpine', '9.6', etc. To the right of the description tab is a box showing an average rating of 5 stars from 13 ratings and a 'Add Product Review' form.

Fig. 25: <https://store.docker.com/images/postgres>

23.7.1.3 What is PostgreSQL ?

See also:

- <https://en.wikipedia.org/wiki/PostgreSQL>

PostgreSQL, often simply “Postgres”, is an object-relational database management system (ORDBMS) with an emphasis on extensibility and standards-compliance.

As a database server, its primary function is to store data, securely and supporting best practices, and retrieve it later, as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet).

It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

Recent versions also provide replication of the database itself for security and scalability.

PostgreSQL implements the majority of the SQL:2011 standard, is ACID-compliant and transactional (including most DDL statements) avoiding locking issues using multiversion concurrency control (MVCC), provides immunity to dirty reads and full serializability; handles complex SQL queries using many indexing methods that are not available in other databases; has updateable views and materialized views, triggers, foreign keys; supports functions and stored procedures, and other expandability, and has a large number of extensions written by third parties. In addition to the possibility of working with the major proprietary and open source databases, PostgreSQL supports migration from them, by its extensive standard SQL support and available migration tools. And if proprietary extensions had been used, by its extensibility that can emulate many through some built-in and third-party open source compatibility extensions, such as for Oracle.

23.7.1.4 Environment Variables

The PostgreSQL image uses several environment variables which are easy to miss. While none of the variables are required, they may significantly aid you in using the image.

23.7.1.4.1 POSTGRES_PASSWORD

This environment variable is recommended for you to use the PostgreSQL image. This environment variable sets the superuser password for PostgreSQL. The default superuser is defined by the POSTGRES_USER environment variable. In the above example, it is being set to “mysecretpassword”.

Note 1: The PostgreSQL image sets up trust authentication locally so you may notice a password is not required when connecting from localhost (inside the same container). However, a password will be required if connecting from a different host/container.

Note 2: This variable defines the superuser password in the PostgreSQL instance, as set by the initdb script during initial container startup. It has no effect on the PGPASSWORD environment variable that may be used by the psql client at runtime, as described at

<https://www.postgresql.org/docs/10/static/libpq-envvars.html>. PGPASSWORD, if used, will be specified as a separate environment variable.

23.7.1.4.2 POSTGRES_USER

This optional environment variable is used in conjunction with POSTGRES_PASSWORD to set a user and its password. This variable will create the specified user with superuser power and a database with the same name. If it is not specified, then the default user of postgres will be used.

23.7.1.4.3 PGDATA

This optional environment variable can be used to define another location - like a subdirectory - for the database files. The default is /var/lib/postgresql/data, but if the data volume you're using is a fs mountpoint (like with GCE persistent disks), Postgres initdb recommends a subdirectory (for example /var/lib/postgresql/data/pgdata) be created to contain the data.

23.7.1.4.4 POSTGRES_DB

This optional environment variable can be used to define a different name for the default database that is created when the image is first started. If it is not specified, then the value of POSTGRES_USER will be used. POSTGRES_INITDB_ARGS

This optional environment variable can be used to send arguments to postgres initdb. The value is a space separated string of arguments as postgres initdb would expect them. This is useful for adding functionality like data page checksums: -e POSTGRES_INITDB_ARGS="--data-checksums".

23.7.1.4.5 POSTGRES_INITDB_WALDIR

This optional environment variable can be used to define another location for the Postgres transaction log. By default the transaction log is stored in a subdirectory of the main Postgres data folder (PGDATA). Sometimes it can be desirable to store the transaction log in a different directory which may be backed by storage with different performance or reliability characteristics.

Note: on PostgreSQL 9.x, this variable is POSTGRES_INITDB_XLOGDIR (reflecting the changed name of the --xlogdir flag to --waldir in PostgreSQL 10+).

23.7.1.5 Docker Secrets

As an alternative to passing sensitive information via environment variables, _FILE may be appended to the previously listed environment variables, causing the initialization script to load the values for those variables from files present in the container. In particular, this can be used to load passwords from Docker secrets stored in /run/secrets/<secret_name> files. For example:

```
$ docker run --name some-postgres -e POSTGRES_PASSWORD_FILE=/run/secrets/postgres-
→passwd -d postgres
```

Currently, this is only supported for POSTGRES_INITDB_ARGS, POSTGRES_PASSWORD, POSTGRES_USER, and POSTGRES_DB

23.7.1.6 How to extend this image

If you would like to do additional initialization in an image derived from this one, add one or more *.sql, .sql.gz, or .sh scripts under /docker-entrypoint-initdb.d (creating the directory if necessary).

After the entrypoint calls initdb to create the default postgres user and database, it will run any .sql files and source any .sh scripts found in that directory to do further initialization before starting the service.

For example, to add an additional user and database, add the following to /docker-entrypoint-initdb.d/init-user-db.sh:

```
#!/bin/bash
set -e

psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
    CREATE USER docker;
    CREATE DATABASE docker;
    GRANT ALL PRIVILEGES ON DATABASE docker TO docker;
EOSQL
```

These initialization files will be executed in sorted name order as defined by the current locale, which defaults to **en_US.utf8**.

Any .sql files will be executed by POSTGRES_USER, which defaults to the postgres superuser.

It is recommended that any psql commands that are run inside of a .sh script be executed as POSTGRES_USER by using the –username “\$POSTGRES_USER” flag. This user will be able to connect without a password due to the presence of trust authentication for Unix socket connections made inside the container.

Additionally, as of docker-library/postgres#253, these initialization scripts are run as the postgres user (or as the “semi-arbitrary user” specified with the –user flag to docker run; see the section titled “Arbitrary –user Notes” for more details).

23.7.1.6.1 Extends with a Dockerfile

You can also **extend the image with a simple Dockerfile** to set a different locale. The following example will set the default locale to de_DE.utf8:

```
FROM postgres:9.4
RUN localedef -i de_DE -c -f UTF-8 -A /usr/share/locale/locale.alias de_DE.UTF-8
ENV LANG de_DE.utf8
```

Since database initialization only happens on container startup, this allows us to set the language before it is created.

23.7.1.7 docker-compose up

```
FROM postgres:10.1
RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
    docker\tutoriels\postgresql> docker-compose up
```

```
WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All
containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Building db
Step 1/3 : FROM postgres:10.1
10.1: Pulling from library/postgres
723254a2c089: Pull complete
39ec0e6c372c: Pull complete
```

(continues on next page)

(continued from previous page)

```

ba1542fb91f3: Pull complete
c7195e642388: Pull complete
95424deca6a2: Pull complete
2d7d4b3a4ce2: Pull complete
fbde41d4a8cc: Pull complete
880120b92add: Pull complete
9a217c784089: Pull complete
d581543fe8e7: Pull complete
e5eff8940bb0: Pull complete
462d60a56b09: Pull complete
135fa6b9c139: Pull complete
Digest: sha256:3f4441460029e12905a5d447a3549ae2ac13323d045391b0cb0cf8b48ea17463
Status: Downloaded newer image for postgres:10.1
--> ec61d13c8566
Step 2/3 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.
--> UTF-8
--> Running in 18aa6161e381
Removing intermediate container 18aa6161e381
--> a20322020edd
Step 3/3 : ENV LANG fr_FR.utf8
--> Running in 0245352c15af
Removing intermediate container 0245352c15af
--> b738f47d14a3
Successfully built b738f47d14a3
Successfully tagged postgres:10.1
WARNING: Image for service db was built because it did not already exist. To rebuild
--> this image you must use `docker-compose build` or `docker-compose up --build`.
Creating container_intranet ... done
Attaching to container_intranet
container_intranet | 2018-01-31 12:09:54.628 UTC [1] LOG:  listening on IPv4 address
--> "0.0.0.0", port 5432
container_intranet | 2018-01-31 12:09:54.628 UTC [1] LOG:  listening on IPv6 address
--> ":::", port 5432
container_intranet | 2018-01-31 12:09:54.839 UTC [1] LOG:  listening on Unix socket "/
--> var/run/postgresql/.s.PGSQL.5432"
container_intranet | 2018-01-31 12:09:55.034 UTC [20] LOG:  database system was shut_
--> down at 2018-01-31 12:03:16 UTC
container_intranet | 2018-01-31 12:09:55.135 UTC [1] LOG:  database system is ready_
--> to accept connections

```

```

PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
--> docker\tutoriels\postgresql> docker exec -ti dda260532cd7 bash

```

```

root@dda260532cd7:/# psql -U postgres
psql (10.1)
Saisissez « help » pour l'aide.

postgres=# \l
                                         Liste des bases de_
--> données
           Nom      | Propriétaire | Encodage | Collationnement | Type caract. |   Droits_
--> d'accès
-----+-----+-----+-----+-----+-----+
-->
-->
postgres  | postgres     | UTF8       | en_US.utf8      | en_US.utf8     | 
template0 | postgres     | UTF8       | en_US.utf8      | en_US.utf8     | =c/postgres 
-->      +

```

(continues on next page)

(continued from previous page)

```

|          |          |          |          |
→postgres=CTc/postgres | postgres | UTF8    | en_US.utf8   | en_US.utf8   | =c/postgres →
→      +               |          |          |          |
→postgres=CTc/postgres |          |          |          |
(3 lignes)

```

23.7.2 Images MariaDB

See also:

- <https://store.docker.com/images/mariadb>
- <https://en.wikipedia.org/wiki/MariaDB>
- <https://fr.wikipedia.org/wiki/MariaDB>

Contents

- *Images MariaDB*
 - *Short Description*
 - *What is MariaDB ?*
 - *How to use this image*



Fig. 26: Le logo MariaDB

23.7.2.1 Short Description

MariaDB is a community-developed fork of MySQL intended to remain free under the GNU GPL

23.7.2.2 What is MariaDB ?

MariaDB is a community-developed fork of the MySQL relational database management system intended to remain free under the GNU GPL.

Supported tags and respective Dockerfile links

- [10.3.4 , 10.3 \(10.3/Dockerfile\)](#)
- [10.2.12 , 10.2 , 10 , latest \(10.2/Dockerfile\)](#)
- [10.1.30 , 10.1 \(10.1/Dockerfile\)](#)
- [10.0.33 , 10.0 \(10.0/Dockerfile\)](#)
- [5.5.58 , 5.5 , 5 \(5.5/Dockerfile\)](#)

Fig. 27: <https://store.docker.com/images/mariadb>

Being a fork of a leading open source software system, it is notable for being led by the original developers of MySQL, who forked it due to concerns over its acquisition by Oracle.

Contributors are required to share their copyright with the MariaDB Foundation.

The intent is also to maintain high compatibility with MySQL, ensuring a “drop-in” replacement capability with library binary equivalency and exact matching with MySQL APIs and commands. It includes the XtraDB storage engine for replacing InnoDB, as well as a new storage engine, Aria, that intends to be both a transactional and non-transactional engine perhaps even included in future versions of MySQL.

23.7.2.3 How to use this image

Start a mariadb server instance

Starting a MariaDB instance is simple:

```
$ docker run --name some-mariadb -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mariadb:tag
```

where some-mariadb is the name you want to assign to your container, my-secret-pw is the password to be set for the MySQL root user and tag is the tag specifying the MySQL version you want.

See the list above for relevant tags. Connect to MySQL from an application in another Docker container

Since MariaDB is intended as a drop-in replacement for MySQL, it can be used with many applications.

This image exposes the standard MySQL port (3306), so container linking makes the MySQL instance available to other application containers. Start your application container like this in order to link it to the MySQL container:

```
$ docker run --name some-app --link some-mariadb:mysql -d application-that-uses-mysql
```

23.7.3 Docker sybase

See also:

- <https://github.com/cbsan/docker-sybase>
- <https://github.com/search?utf8=%E2%9C%93&q=docker+sybase&type=>

Contents

- [*Docker sybase*](#)

23.8 Images message queue

23.8.1 Images rabbitmq

See also:

- <https://store.docker.com/images/rabbitmq>
- https://hub.docker.com/_/rabbitmq/
- <https://en.wikipedia.org/wiki/RabbitMQ>

Contents

- [*Images rabbitmq*](#)
 - [*What is RabbitMQ ?*](#)
 - [*Rabbitmq and celery*](#)



Fig. 28: Le logo de rabbitmq

23.8.1.1 What is RabbitMQ ?

RabbitMQ is open source message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP).

The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.

23.8.1.2 Rabbitmq and celery

See also:

- <http://docs.celeryproject.org/en/latest/getting-started/brokers/rabbitmq.html>

23.9 Images outils collaboratifs

23.9.1 Images Gitlab community edition

See also:

- <https://store.docker.com/images/gitlab-community-edition>
- <https://about.gitlab.com/features/>

Contents

- *Images Gitlab community edition*
 - *Short Description*



Fig. 29: Le logo Gitlab

23.9.1.1 Short Description

GitLab includes Git repository management, issue tracking, code review, an IDE, activity streams, wikis, and more.

Open source collaboration and source control management: code, test, and deploy together! More details on features can be found on <https://about.gitlab.com/features/>

23.9.2 Images Redmine

See also:

- <https://store.docker.com/images/redmine>

Contents

- *Images Redmine*
 - *Short Description*



Fig. 30: Le logo redmine

23.9.2.1 Short Description

Redmine is a flexible project management web application written using Ruby on Rails framework.

23.9.3 Images Wordpress

See also:

- <https://store.docker.com/images/wordpress>

Contents

- *Images Wordpress*
 - *Short Description*

23.9.3.1 Short Description

The WordPress rich content management system can utilize plugins, widgets, and themes.



Fig. 31: Le logo redmine

23.10 Images “documentation”

23.10.1 Images MiKTeX

See also:

- <https://store.docker.com/community/images/miktex/miktex>
- <https://github.com/MiKTeX/docker-miktex>
- <https://en.wikipedia.org/wiki/MiKTeX>

Contents

- *Images MiKTeX*
 - *Short Description*

23.10.1.1 Short Description

MiKTeX is a distribution of the TeX/LaTeX typesetting system for Microsoft Windows. It also contains a set of related programs.

MiKTeX provides the tools necessary to prepare documents using the TeX/LaTeX markup language, as well as a simple tex editor: TeXworks.

The name comes from Christian Schenk's login: MiK for Micro-Kid.

23.11 Images outils scientifiques

23.11.1 Images Anaconda3

See also:

- <https://docs.anaconda.com/anaconda/user-guide/tasks/integration/docker>
- <https://hub.docker.com/r/continuumio/>
- <https://hub.docker.com/r/continuumio/anaconda3/>
- <https://github.com/ContinuumIO/docker-images>
- <https://docs.anaconda.com/anaconda/glossary>

Contents

- *Images Anaconda3*
 - *Short Description*
 - *Usage*

23.11.1.1 Short Description

Powerful and flexible python distribution.

23.11.1.2 Usage

You can download and run this image using the following commands:

```
C:\Tmp>docker pull continuumio/anaconda3
```

```
Using default tag: latest
latest: Pulling from continuumio/anaconda3
85b1f47fba49: Pull complete
f4070d96116d: Pull complete
8b1142e4866d: Pull complete
924a14505c9a: Pull complete
Digest: sha256:c6fb10532fe2efac2f61bd4941896b917ad7b7f197bda9bdd3943aee434d281
Status: Downloaded newer image for continuumio/anaconda3:latest
```

```
C:\Tmp>docker run -i -t continuumio/anaconda3 /bin/bash
```

```
root@8ffcdde2f70f6:/# uname -a
```

```
Linux 8ffcdde2f70f6 4.9.60-linuxkit-aufs #1 SMP Mon Nov 6 16:00:12 UTC 2017 x86_64 GNU/
↳ Linux
```

```
root@8ffcdde2f70f6:/# which python
```



Fig. 32: Le logo Continuumio

```
/opt/conda/bin/python
```

```
root@8fffcde2f70f6:/# python
```

```
Python 3.6.3 |Anaconda, Inc.| (default, Oct 13 2017, 12:02:49)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

23.12 Images apprentissage

23.12.1 Image dockersamples/static-site

See also:

- <https://hub.docker.com/r/dockersamples/static-site/>

Contents

- *Image dockersamples/static-site*

23.12.2 Image hello world

See also:

- https://hub.docker.com/_/hello-world/
- <https://store.docker.com/images/hello-world>
- <https://github.com/docker/labs/blob/master/beginner/chapters/setup.md>

Contents

- *Image hello world*
 - *Short Description*

23.12.2.1 Short Description

Hello World! (an example of minimal Dockerization).

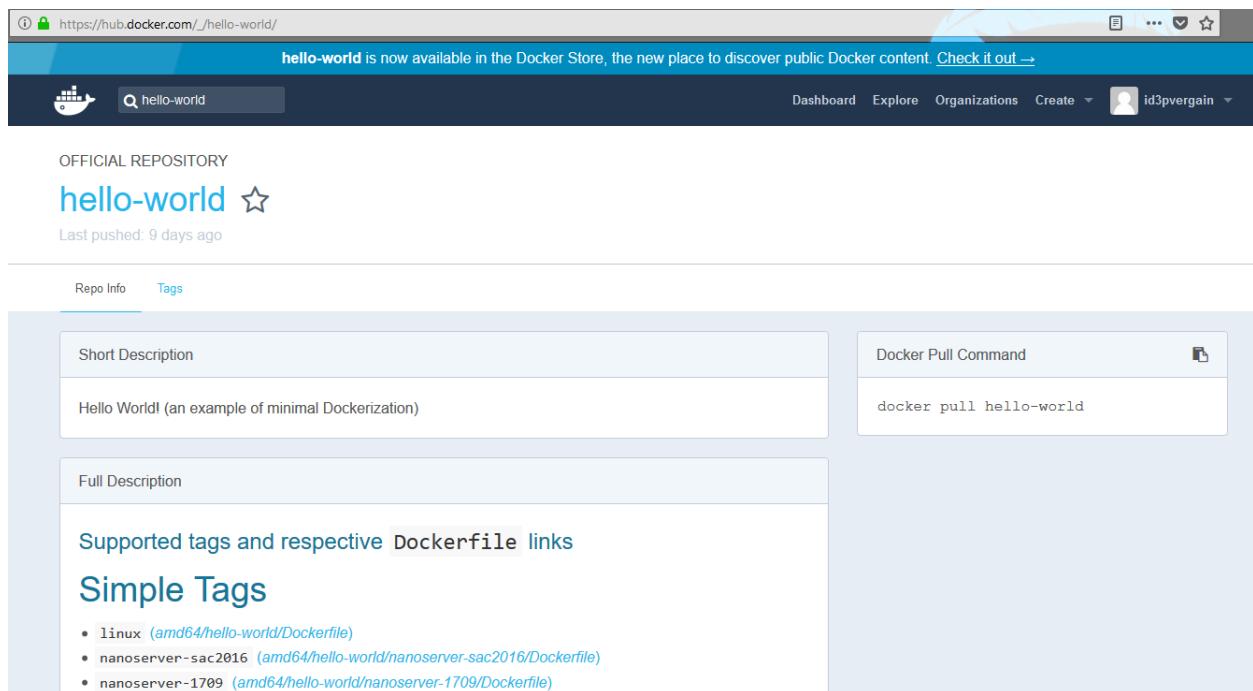


Fig. 33: https://hub.docker.com/_/hello-world/

CHAPTER 24

Tutoriels Docker

See also:

- <https://docs.docker.com/>
- <https://github.com/docker/labs/>
- <https://twitter.com/Docker/lists/docker-captains/members>
- <https://github.com/jpetazzo/container.training>
- <https://hackr.io/tutorials/learn-docker>

24.1 Avril 2018 container training from Jérôme Petazzoni

See also:

- <https://avril2018.container.training>
- *Les conseils et formations de Jérôme Petazzoni*
- <https://github.com/jpetazzo/container.training/graphs/contributors>
- <https://github.com/jpetazzo/container.training>

24.1.1 Intro Avril 2018

See also:

- <https://github.com/jpetazzo/container.training>
- <https://avril2018.container.training/intro.yml.html#1>
- <https://avril2018.container.training/intro.yml.html#16>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.1.1 Overview

See also:

- <https://github.com/jpetazzo/container.training>
- <https://avril2018.container.training/intro.yml.html#1>
- <https://avril2018.container.training/intro.yml.html#16>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Overview*
 - *A brief introduction*
 - *About these slides*
 - *Docker 30,000ft overview*
 - *OK... Why the buzz around containers ?*
 - *Deployment becomes very complex*
 - *Results*
 - *Escape dependency hell*
 - *On-board developers and contributors rapidly*
 - *Implement reliable CI easily*
 - *Use container images as build artefacts*
 - *Decouple “plumbing” from application logic*
 - *Formats and APIs, before Docker*
 - *Formats and APIs, after Docker*
 - *Shipping, before Docker*
 - *Shipping, after Docker*
 - *Example*
 - *Devs vs Ops, before Docker*
 - *Devs vs Ops, after Docker*

24.1.1.1 A brief introduction

See also:

- <https://avril2018.container.training/intro.yml.html#3>
- <https://github.com/jpetazzo/container.training/graphs/contributors>

This was initially written to support in-person, instructor-led workshops and tutorials

These materials are maintained by Jérôme Petazzoni and [multiple contributors](#)

You can also follow along on your own, at your own pace

We included as much information as possible in these slides

We recommend having a mentor to help you ...

... Or be comfortable spending some time reading the Docker documentation ...

... And looking for answers in the Docker forums, StackOverflow, and other outlets

24.1.1.1.2 About these slides

See also:

- <https://avril2018.container.training/intro.yml.html#5>

All the content is available in a public GitHub repository:

- <https://github.com/jpetazzo/container.training>

You can get updated “builds” of the slides there:

- <http://container.training/>

Typos? Mistakes? Questions? Feel free to hover over the bottom of the slide ...

24.1.1.1.3 Docker 30,000ft overview

See also:

- <https://avril2018.container.training/intro.yml.html#17>

In this lesson, we will learn about:

- Why containers (non-technical elevator pitch)
- Why containers (technical elevator pitch)
- How Docker helps us to build, ship, and run
- The history of containers

We won't actually run Docker or containers in this chapter (yet!).

Don't worry, we will get to that fast enough !

24.1.1.1.4 OK... Why the buzz around containers ?

See also:

- <https://avril2018.container.training/intro.yml.html#19>

The software industry has changed

Before:

- monolithic applications
- long development cycles
- single environment
- slowly scaling up

Now:

- decoupled services
- fast, iterative improvements
- multiple environments
- quickly scaling out

24.1.1.5 Deployment becomes very complex

See also:

- <https://avril2018.container.training/intro.yml.html#20>

Many different stacks:

- languages
- frameworks
- databases

Many different targets:

- individual development environments
- pre-production, QA, staging...
- production: on prem, cloud, hybrid

24.1.1.6 Results

See also:

- <https://avril2018.container.training/intro.yml.html#28>
- Dev-to-prod reduced from 9 months to 15 minutes (ING)
- Continuous integration job time reduced by more than 60% (BBC)
- Deploy 100 times a day instead of once a week (GILT)
- 70% infrastructure consolidation (MetLife)
- 60% infrastructure consolidation (Intesa Sanpaolo)
- 14x application density; 60% of legacy datacenter migrated in 4 months (GE Appliances)
- etc.

24.1.1.7 Escape dependency hell

See also:

- <https://avril2018.container.training/intro.yml.html#30>
- Write installation instructions into an INSTALL.txt file
- Using this file, write an install.sh script that works for you
- Turn this file into a Dockerfile, test it on your machine
- If the Dockerfile builds on your machine, it will build anywhere

- Rejoice as you escape dependency hell and “works on my machine”
- Never again “worked in dev - ops problem now!”

24.1.1.8 On-board developers and contributors rapidly

See also:

- <https://avril2018.container.training/intro.yml.html#31>
- Write Dockerfiles for your application component
- Use pre-made images from the Docker Hub (mysql, redis...)
- Describe your stack with a Compose file
- On-board somebody with two commands:

```
git clone ...
docker-compose up
```

With this, you can create development, integration, QA environments in minutes !

24.1.1.9 Implement reliable CI easily

See also:

- <https://avril2018.container.training/intro.yml.html#32>
- Build test environment with a Dockerfile or Compose file
- For each test run, stage up a new container or stack
- Each run is now in a clean environment
- No pollution from previous tests

Way faster and cheaper than creating VMs each time!

24.1.1.10 Use container images as build artefacts

See also:

- <https://avril2018.container.training/intro.yml.html#33>
- Build your app from Dockerfiles
- Store the resulting images in a registry
- Keep them forever (or as long as necessary)
- Test those images in QA, CI, integration...
- Run the same images in production
- Something goes wrong? Rollback to previous image
- Investigating old regression? Old image has your back!

Images contain all the libraries, dependencies, etc. needed to run the app.

24.1.1.11 Decouple “plumbing” from application logic

See also:

- <https://avril2018.container.training/intro.yml.html#34>
- Write your code to connect to named services (“db”, “api”...)
- Use Compose to start your stack
- Docker will setup per-container DNS resolver for those names
- You can now scale, add load balancers, replication ... without changing your code

Note: this is not covered in this intro level workshop !

24.1.1.12 Formats and APIs, before Docker

See also:

- <https://avril2018.container.training/intro.yml.html#36>
- No standardized exchange format. (No, a rootfs tarball is not a format!)
- Containers are hard to use for developers. (Where’s the equivalent of docker run debian?)
- As a result, they are hidden from the end users.
- No re-usable components, APIs, tools. (At best: VM abstractions, e.g. libvirt.)

Analogy:

- Shipping containers are not just steel boxes.
- They are steel boxes that are a standard size, with the same hooks and holes.

24.1.1.13 Formats and APIs, after Docker

See also:

- <https://avril2018.container.training/intro.yml.html#37>
- Standardize the container format, because containers were not portable.
- Make containers easy to use for developers.
- Emphasis on re-usable components, APIs, ecosystem of standard tools.
- Improvement over ad-hoc, in-house, specific tools.

24.1.1.14 Shipping, before Docker

See also:

- <https://avril2018.container.training/intro.yml.html#38>
- Ship packages: deb, rpm, gem, jar, homebrew...
- Dependency hell.
- “Works on my machine.”
- Base deployment often done from scratch (debootstrap...) and unreliable.

24.1.1.15 Shipping, after Docker

See also:

- <https://avril2018.container.training/intro.yml.html#39>
- Ship container images with all their dependencies.
- Images are bigger, but they are broken down into layers.
- Only ship layers that have changed.
- Save disk, network, memory usage.

24.1.1.16 Example

See also:

- <https://avril2018.container.training/intro.yml.html#40>

Layers:

- CentOS
- JRE
- Tomcat
- Dependencies
- Application JAR
- Configuration

24.1.1.17 Devs vs Ops, before Docker

See also:

- <https://avril2018.container.training/intro.yml.html#41>
- Drop a tarball (or a commit hash) with instructions.
- Dev environment very different from production.
- Ops don't always have a dev environment themselves ...
- ... and when they do, it can differ from the devs' .
- Ops have to sort out differences and make it work ...
- ... or bounce it back to devs.
- Shipping code causes frictions and delays.

24.1.1.18 Devs vs Ops, after Docker

See also:

- <https://avril2018.container.training/intro.yml.html#42>
- Drop a container image or a Compose file.
- Ops can always run that container image.

- Ops can always run that Compose file.
- Ops still have to adapt to prod environment, but at least they have a reference point.
- Ops have tools allowing to use the same image in dev and prod.
- Devs can be empowered to make releases themselves more easily.

24.1.1.2 History of containers ... and Docker

See also:

- <https://avril2018.container.training/intro.yml.html#44>
- <https://avril2018.container.training/intro.yml.html#1>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *History of containers ... and Docker*
 - *First experimentations*
 - *The VPS age (until 2007-2008)*
 - *Containers = cheaper than VMs*
 - *The PAAS period (2008-2013)*
 - *Containers = easier than VMs*
 - *First public release of Docker*
 - *Docker early days (2013-2014)*
 - *First users of Docker*
 - *Positive feedback loop*
 - *Maturity (2015-2016)*
 - *Docker becomes an industry standard*
 - *Docker becomes a platform*

24.1.1.2.1 First experimentations

See also:

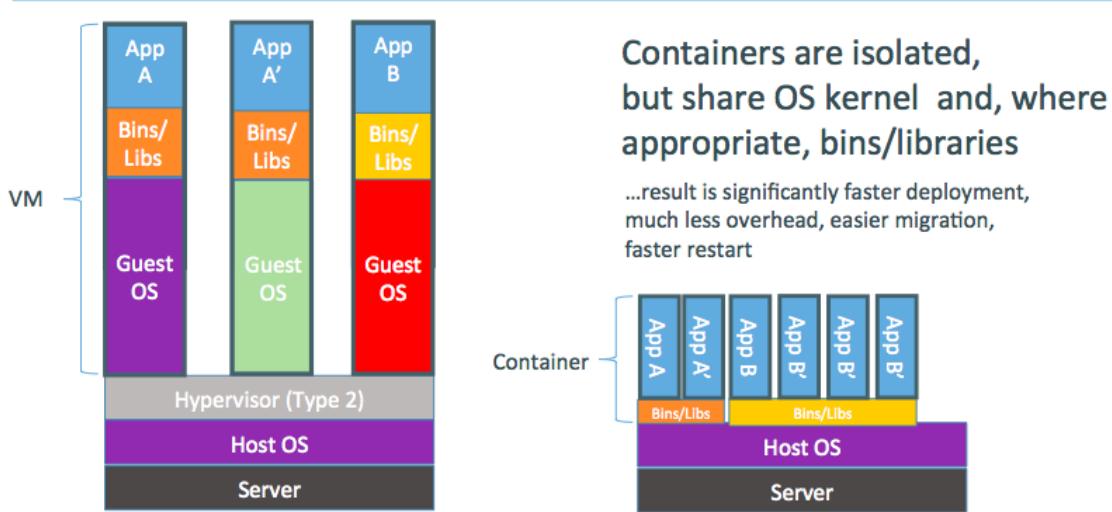
- <https://avril2018.container.training/intro.yml.html#46>
- IBM VM/370 (1972)
- Linux VServers (2001)
- Solaris Containers (2004)
- FreeBSD jails (1999)

Containers have been around for a very long time indeed.

24.1.1.2.2 The VPS age (until 2007-2008)

See also:

- <https://avril2018.container.training/intro.yml.html#47>



24.1.1.2.3 Containers = cheaper than VMs

See also:

- <https://avril2018.container.training/intro.yml.html#48>
- Users: hosting providers.
- Highly specialized audience with strong ops culture.

24.1.1.2.4 The PAAS period (2008-2013)

See also:

- <https://avril2018.container.training/intro.yml.html#49>

24.1.1.2.5 Containers = easier than VMs

See also:

- <https://avril2018.container.training/intro.yml.html#50>
- I can't speak for Heroku, but containers were (one of) dotCloud's secret weapon
- dotCloud was operating a PaaS, using a custom container engine.
- This engine was based on OpenVZ (and later, LXC) and AUFS.
- It started (circa 2008) as a single Python script.

The screenshot shows the Heroku website's features section. It includes four main items:

- Instantly Live**: When you create or import an app with Heroku, it's already live on the web. No configuration or deployment necessary.
[Watch Demo ▶](#)
- Create and Edit Online**: Edit all of your code and data right in your browser. You can access it anywhere and there's nothing to install.
[Watch Demo ▶](#)
- Share and Collaborate**: Make your app public to the web or private to users you specify. Add collaborators who can edit the app with you.
[Watch Demo ▶](#)
- Import & Export**: Easily import and export your app's code, schema, and data at any time with just one click.

[Sign up](#) for our limited beta and we'll send you an invitation as soon as possible.

- By 2012, the engine had multiple (~10) Python components.
- (and ~100 other micro-services!)
- End of 2012, dotCloud refactors this container engine.

The codename for this project is *Docker*.

24.1.1.2.6 First public release of Docker

See also:

- <https://avril2018.container.training/intro.yml.html#51>
- March 2013, PyCon, Santa Clara: “Docker” is shown to a public audience for the first time.
- It is released with an open source license.
- Very positive reactions and feedback!
- The dotCloud team progressively shifts to Docker development.
- The same year, dotCloud changes name to Docker.
- In 2014, the PaaS activity is sold.

24.1.1.2.7 Docker early days (2013-2014)

See also:

- <https://avril2018.container.training/intro.yml.html#52>

24.1.1.2.8 First users of Docker

See also:

- <https://avril2018.container.training/intro.yml.html#53>
- PAAS builders (Flynn, Dokku, Tsuru, Deis...)
- PAAS users (those big enough to justify building their own)
- CI platforms
- developers, developers, developers, developers

24.1.1.2.9 Positive feedback loop

See also:

- <https://avril2018.container.training/intro.yml.html#54>
- In 2013, the technology under containers (cgroups, namespaces, copy-on-write storage...) had many blind spots.
- The growing popularity of Docker and containers exposed many bugs.
- As a result, those bugs were fixed, resulting in better stability for containers.
- Any decent hosting/cloud provider can run containers today.
- Containers become a great tool to deploy/move workloads to/from on-prem/cloud.

24.1.1.2.10 Maturity (2015-2016)

See also:

- <https://avril2018.container.training/intro.yml.html#55>

24.1.1.2.11 Docker becomes an industry standard

See also:

- <https://avril2018.container.training/intro.yml.html#56>
- Docker reaches the symbolic 1.0 milestone.
- Existing systems like Mesos and Cloud Foundry add Docker support.
- Standardization around the OCI (Open Containers Initiative).
- Other container engines are developed.
- Creation of the CNCF (Cloud Native Computing Foundation).

24.1.1.2.12 Docker becomes a platform

See also:

- <https://avril2018.container.training/intro.yml.html#56>

The initial container engine is now known as **Docker Engine**

Other tools are added:

- Docker Compose (formerly “Fig”)
- Docker Machine
- Docker Swarm
- Kitematic
- Docker Cloud (formerly “Tutum”)
- Docker Datacenter
- etc.

Docker Inc. launches commercial offers.

24.1.2 Chapter1 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#7>
- <https://avril2018.container.training/intro.yml.html#16>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.3 Chapter2 Avril 2018 container training

See also:

- <https://avril2018.container.training/intro.yml.html#8>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.3.1 Our first containers

See also:

- <https://avril2018.container.training/intro.yml.html#79>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Our first containers*
 - *Hello World*
 - *Starting another container*

24.1.3.1.1 Hello World

See also:

- <https://avril2018.container.training/intro.yml.html#82>
- `docker run`

```
# docker run busybox echo hello world
```

```
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
07a152489297: Pull complete
Digest: sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f41046e0f37d47
Status: Downloaded newer image for busybox:latest
hello world
```

24.1.3.1.2 Starting another container

See also:

- <https://avril2018.container.training/intro.yml.html#92>

24.1.3.2 Backgroud containers

Contents

- *Backgroud containers*
 - *Objectives*
 - *A non-interactive container*
 - *Run a container in the background*
 - *List running containers*
 - *View only the IDs of the containers*
 - *Combining flags*
 - *View the logs of a container*
 - *View only the tail of the logs*
 - *Follow the logs in real time*
 - *Stop our container*
 - *Stopping our containers*
 - *Killing the remaining containers*

24.1.3.2.1 Objectives

Our first containers were interactive.

We will now see how to:

- Run a non-interactive container.
- Run a container in the background.
- List running containers.
- Check the logs of a container.
- Stop a container.
- List stopped containers.

24.1.3.2.2 A non-interactive container

See also:

- <https://avril2018.container.training/intro.yml.html#97>

```
$ docker run jpetazzo/clock
```

```
Unable to find image 'jpetazzo/clock:latest' locally
latest: Pulling from jpetazzo/clock
a3ed95caeb02: Pull complete
1db09adb5ddd: Pull complete
Digest: sha256:446edaa1594798d89ee2a93f660161b265db91b026491e4671c14371eff5eea0
Status: Downloaded newer image for jpetazzo/clock:latest
Wed May 30 08:34:23 UTC 2018
Wed May 30 08:34:24 UTC 2018
Wed May 30 08:34:25 UTC 2018
Wed May 30 08:34:26 UTC 2018
Wed May 30 08:34:27 UTC 2018
Wed May 30 08:34:28 UTC 2018
Wed May 30 08:34:29 UTC 2018
```

24.1.3.2.3 Run a container in the background

See also:

- <https://avril2018.container.training/intro.yml.html#98>

Containers can be started in the background, with the -d flag (daemon mode)

```
$ docker run -d jpetazzo/clock
```

```
36935b2a967fd69c7fa23788e00855baa1896cc4af111fb78b9cfcc70a4d409c
```

- We don't see the output of the container.
- But don't worry: Docker collects that output and logs it!
- Docker gives us the ID of the container.

24.1.3.2.4 List running containers

See also:

- <https://avril2018.container.training/intro.yml.html#99>
- `docker ps`

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
36935b2a967f	jpetazzo/clock	"/bin/sh -c 'while d..."	3 minutes ago
Up 3 minutes		lucid_kapitsa	

Docker tells us:

- The (truncated) ID of our container.
- The image used to start the container.
- That our container has been running (Up) for a couple of minutes.
- Other information (COMMAND, PORTS, NAMES) that we will explain later

24.1.3.2.5 View only the IDs of the containers

See also:

- <https://avril2018.container.training/intro.yml.html#102>
- `docker ps`

Many Docker commands will work on container IDs: `docker stop`, `docker rm...`

If we want to list only the IDs of our containers (without the other columns or the header line), we can use the `-q` (“Quiet”, “Quick”) flag:

```
docker ps -q
```

24.1.3.2.6 Combining flags

See also:

- <https://avril2018.container.training/intro.yml.html#102>
- `docker ps`

We can combine `-l` and `-q` to see only the ID of the last container started:

```
$ docker ps -lq
```

```
3c181d8dbc4c
```

At a first glance, it looks like this would be particularly useful in scripts.

However, if we want to start a container and get its ID in a reliable way, it is better to use `docker run -d`, which we will cover in a bit.

24.1.3.2.7 View the logs of a container

See also:

- <https://avril2018.container.training/intro.yml.html#104>
- *docker logs*

We told you that Docker was logging the container output.

Let's see that now.

```
docker logs 3c18
```

```
Wed May 30 08:49:15 UTC 2018
Wed May 30 08:49:16 UTC 2018
Wed May 30 08:49:17 UTC 2018
Wed May 30 08:49:18 UTC 2018
Wed May 30 08:49:19 UTC 2018
Wed May 30 08:49:20 UTC 2018
Wed May 30 08:49:21 UTC 2018
Wed May 30 08:49:22 UTC 2018
Wed May 30 08:49:23 UTC 2018
Wed May 30 08:49:24 UTC 2018
Wed May 30 08:49:25 UTC 2018
Wed May 30 08:49:26 UTC 2018
Wed May 30 08:49:27 UTC 2018
```

24.1.3.2.8 View only the tail of the logs

See also:

- <https://avril2018.container.training/intro.yml.html#104>
- *docker logs*

To avoid being spammed with eleventy pages of output, we can use the `--tail` option:

```
$ docker logs 3c18 --tail 3
```

```
Wed May 30 09:02:29 UTC 2018
Wed May 30 09:02:30 UTC 2018
Wed May 30 09:02:31 UTC 2018
```

The parameter is the number of lines that we want to see.

24.1.3.2.9 Follow the logs in real time

See also:

- <https://avril2018.container.training/intro.yml.html#106>
- *docker logs*

Just like with the standard UNIX command `tail -f`, we can follow the logs of our container:

```
$ docker logs --follow 3c18 --tail 1
```

```
Wed May 30 09:07:11 UTC 2018
Wed May 30 09:07:12 UTC 2018
Wed May 30 09:07:13 UTC 2018
Wed May 30 09:07:14 UTC 2018
Wed May 30 09:07:15 UTC 2018
```

- This will display the last line in the log file.
- Then, it will continue to display the logs in real time.
- Use ^C to exit.

24.1.3.2.10 Stop our container

See also:

- <https://avril2018.container.training/intro.yml.html#107>
- *docker stop*

There are two ways we can terminate our detached container.

- Killing it using the docker kill command.
- Stopping it using the docker stop command.

The first one stops the container immediately, by using the KILL signal.

The second one is more graceful. It sends a TERM signal, and after 10 seconds, if the container has not stopped, it sends KILL.

Reminder: the KILL signal cannot be intercepted, and will forcibly terminate the container.

24.1.3.2.11 Stopping our containers

See also:

- <https://avril2018.container.training/intro.yml.html#108>
- *docker stop*

Let's stop one of those containers:

```
$ docker stop 3c1
```

```
3c1
```

This will take 10 seconds:

- Docker sends the TERM signal;
- the container doesn't react to this signal (it's a simple Shell script with no special signal handling);
- 10 seconds later, since the container is still running, Docker sends the KILL signal;
- this terminates the container.

24.1.3.2.12 Killing the remaining containers

See also:

- <https://avril2018.container.training/intro.yml.html#109>
- *docker stop*

Let's be less patient with the two other containers:

```
$ docker stop 1fe 369
```

```
1fe  
369
```

The stop and kill commands can take multiple container IDs.

Those containers will be terminated immediately (without the 10 seconds delay).

Let's check that our containers don't show up anymore:

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	
STATUS	PORTS	NAMES		

24.1.3.3 Restarting and attaching to containers

Contents

- *Restarting and attaching to containers*
 - *Introduction*
 - *Background and foreground*
 - *Detaching from a container*
 - *Specifying a custom detach sequence*
 - *Attaching to a container*
 - *Detaching from non-interactive containers*
 - *Restarting a container*

24.1.3.3.1 Introduction

See also:

- <https://avril2018.container.training/intro.yml.html#112>

We have started containers in the foreground, and in the background.

In this chapter, we will see how to:

- Put a container in the background.

- Attach to a background container to bring it to the foreground.
- Restart a stopped container.

24.1.3.3.2 Background and foreground

See also:

- <https://avril2018.container.training/intro.yml.html#114>

The distinction between foreground and background containers is arbitrary.

From Docker's point of view, all containers are the same.

All containers run the same way, whether there is a client attached to them or not.

It is always possible to detach from a container, and to reattach to a container.

Analogy: attaching to a container is like plugging a keyboard and screen to a physical server.

24.1.3.3.3 Detaching from a container

See also:

- <https://avril2018.container.training/intro.yml.html#115>

If you have started an interactive container (with option -it), you can detach from it.

The “detach” sequence is ^P^Q.

Otherwise you can detach by killing the Docker client.

(But not by hitting ^C, as this would deliver SIGINT to the container.)

What does -it stand for?

- -t means “allocate a terminal.”
- -i means “connect stdin to the terminal.”

24.1.3.3.4 Specifying a custom detach sequence

See also:

- <https://avril2018.container.training/intro.yml.html#116>
- You don't like ^P^Q? No problem!
- You can change the sequence with docker run --detach-keys.
- This can also be passed as a global option to the engine.

Start a container with a custom detach command:

```
$ docker run -ti --detach-keys ctrl-x,x jpetazzo/clock
```

Detach by hitting ^X x. (This is ctrl-x then x, not ctrl-x twice!)

Check that our container is still running:

```
$ docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
16c9e8ab42c3	jpetazzo/clock	"/bin/sh -c 'while d..."	26 seconds ago
Up 24 seconds		unruffled_joliot	

24.1.3.3.5 Attaching to a container

See also:

- <https://avril2018.container.training/intro.yml.html#117>

You can attach to a container

```
$ docker attach <containerID>
```

The container must be running. There can be multiple clients attached to the same container. If you don't specify `-detach-keys` when attaching, it defaults back to `^P^Q`.

Try it on our previous container

```
$ docker attach $(docker ps -lq)
```

Check that `^X x` doesn't work, but `^P ^Q` does.

24.1.3.3.6 Detaching from non-interactive containers

See also:

- <https://avril2018.container.training/intro.yml.html#118>

Warning: if the container was started without `-it...`

- You won't be able to detach with `^P^Q`.
- If you hit `^C`, the signal will be proxied to the container.

Remember: you can always detach by killing the Docker client.

24.1.3.3.7 Restarting a container

When a container has exited, it is in stopped state.

It can then be restarted with the `start` command.

```
$ docker start <yourContainerID>
```

The container will be restarted using the same options you launched it with.

You can re-attach to it if you want to interact with it.

```
$ docker attach <yourContainerID>
```

Use `docker ps -a` to identify the container ID of a previous `jpetazzo/clock` container, and try those commands.

24.1.3.4 Understanding Docker images

See also:

- <https://avril2018.container.training/intro.yml.html#124>

Contents

- *Understanding Docker images*
 - *Objectives*
 - *What is an image ?*
 - *Differences between containers and images*
 - *Object-oriented programming*
 - *Wait a minute*
 - *Creating the first images*
 - *Creating other images*
 - * *docker commit*
 - * *docker build*
 - *Images namespaces*
 - *Root namespace*
 - *User namespace*
 - *Self-Hosted namespace*
 - *How do you store and manage images ?*
 - *Showing current images*
 - *Searching for images*
 - *Downloading images*
 - *Pulling an image*
 - *Image and tags*
 - *When to (not) use tags*
 - * *Don't specify tags*
 - * *Do specify tags*
 - *Section summary*

24.1.3.4.1 Objectives

In this section, we will explain:

- What is an image.
- What is a layer.
- The various image namespaces.

- How to search and download images.
- Image tags and when to use them.

24.1.3.4.2 What is an image ?

See also:

- <https://avril2018.container.training/intro.yml.html#127>

Image = files + metadata

These files form the root filesystem of our container.

The metadata can indicate a number of things, e.g.:

- the author of the image
- the command to execute in the container when starting it
- environment variables to be set
- etc.

Images are made of layers, conceptually stacked on top of each other.

Each layer can add, change, and remove files and/or metadata.

Images can share layers to optimize disk usage, transfer times, and memory use.

24.1.3.4.3 Differences between containers and images

See also:

- <https://avril2018.container.training/intro.yml.html#129>
- An image is a read-only filesystem.
- A container is an encapsulated set of processes running in a read-write copy of that filesystem.
- To optimize container boot time, copy-on-write is used instead of regular copy.
- docker run starts a container from a given image.

Let's give a couple of metaphors to illustrate those concepts.

24.1.3.4.4 Object-oriented programming

See also:

- <https://avril2018.container.training/intro.yml.html#131>
- Images are conceptually similar to classes.
- Layers are conceptually similar to inheritance.
- Containers are conceptually similar to instances.

24.1.3.4.5 Wait a minute

See also:

- <https://avril2018.container.training/intro.yml.html#132>

If an image is read-only, how do we change it?

- We don't.
- We create a new container from that image.
- Then we make changes to that container.
- When we are satisfied with those changes, we transform them into a new layer.
- A new image is created by stacking the new layer on top of the old image.

24.1.3.4.6 Creating the first images

See also:

- <https://avril2018.container.training/intro.yml.html#134>

There is a special empty image called scratch. It allows to build from scratch.

The docker import command loads a tarball into Docker.

- The imported tarball becomes a standalone image.
- That new image has a single layer.

24.1.3.4.7 Creating other images

See also:

- <https://avril2018.container.training/intro.yml.html#135>

docker commit

- Saves all the changes made to a container into a new layer.
- Creates a new image (effectively a copy of the container).

docker build

- Performs a repeatable build sequence.
- This is the preferred method!

We will explain both methods in a moment.

24.1.3.4.8 Images namespaces

See also:

- <https://avril2018.container.training/intro.yml.html#136>

There are three namespaces:

- Official images:
 - e.g. ubuntu, busybox ...
- User (and organizations) images:
 - e.g. jpetazzo/clock
- Self-hosted images:
 - e.g. registry.example.com:5000/my-private/image

Let's explain each of them.

24.1.3.4.9 Root namespace

See also:

- <https://avril2018.container.training/intro.yml.html#137>

The root namespace is for official images. They are put there by Docker Inc., but they are generally authored and maintained by third parties.

Those images include:

- Small, “swiss-army-knife” images like busybox.
- Distro images to be used as bases for your builds, like ubuntu, fedora...
- Ready-to-use components and services, like redis, postgresql...

24.1.3.4.10 User namespace

See also:

- <https://avril2018.container.training/intro.yml.html#138>

The user namespace holds images for Docker Hub users and organizations.

For example:

- jpetazzo/clock

The Docker Hub user is:

- jpetazzo

The image name is:

- clock

24.1.3.4.11 Self-Hosted namespace

See also:

- <https://avril2018.container.training/intro.yml.html#139>

This namespace holds images which are not hosted on Docker Hub, but on third party registries.

They contain the hostname (or IP address), and optionally the port, of the registry server.

For example:

- localhost:5000/wordpress
- **localhost:5000** is the host and port of the registry
- **wordpress** is the name of the image

24.1.3.4.12 How do you store and manage images ?

See also:

- <https://avril2018.container.training/intro.yml.html#140>

Images can be stored:

- On your Docker host.
- In a Docker registry.

You can use the Docker client to download (pull) or upload (push) images.

To be more accurate: you can use the Docker client to tell a Docker Engine to push and pull images to and from a registry.

24.1.3.4.13 Showing current images

See also:

- <https://avril2018.container.training/intro.yml.html#141>

Let's look at what images are on our host now.

docker images				
REPOSITORY	TAG	IMAGE ID	CREATED	
ch4-message-board-app_web	latest	376512737492	21 hours ago	[1]
gdevops/django36_ch4	latest	b44a8c214cdf	22 hours ago	[1]
postgres	10.4	61d053fc271c	4 days ago	[1]
busybox	latest	8c811b4aec35	6 days ago	[1]
python	3.6	29d2f3226daf	3 weeks ago	[1]
ubuntu	latest	452a96d81c30	4 weeks ago	[1]
jpetazzo/clock	latest	12068b93616f	3 years ago	[1]
↳ 2.43MB			(continues on next page)	

(continued from previous page)

24.1.3.4.14 Searching for images

See also:

- <https://avril2018.container.training/intro.yml.html#142>

We cannot list all images on a remote registry, but we can search for a specific keyword:

```
$ docker search marathon
```

NAME	OFFICIAL	DESCRIPTION
→ STARS		AUTOMATED
mesosphere/marathon		A cluster-wide init and control system for Marathon
→ s... 106		[OK]
mesoscloud/marathon		Marathon
→ 31		[OK]
mesosphere/marathon-lb		Script to update haproxy based on marathon
→ s... 22		[OK]
mesosphere/marathon-lb-autoscale		Autoscale your apps on Marathon
→ 5		[OK]
thefactory/marathon		Tagged images of each Mesos Marathon release
→ 4		[OK]
brndnmthws/marathon-lb-autoscale		Marathon-lb autoscale demo
→ 3		[OK]
mesoscloud/haproxy-marathon		[DEPRECATED] Generate HAProxy configuration
→ ... 3		[OK]
f5networks/marathon-asp-ctlr		Official container repository for F5
→ Maratho... 3		
bobrik/marathon-tcp-haproxy		
→ 2		[OK]
tobilg/marathon-slack		Listen to Marathon's Event Bus and send
→ sele... 2		[OK]
f5networksdevel/marathon-bigip-ctlr		Container repository for development images
→ ... 1		
tobilg/gitlab-ci-runner-marathon		A customized Docker image for running
→ scalab... 1		[OK]
eduser25/pg-marathon-watcher		PG Marathon watcher application for Marathon
→ ... 1		
vidazoohub/marathon-rabbit-autoscale		autoscale marathon tasks based on rabbitmq
→ q... 1		[OK]
gettyimages/marathon_exporter		Marathon metrics exporter for Prometheus
→ 0		
skytix/marathon-consul		Consul service registration daemon that
→ moni... 0		
heww/marathon-dns		dns for marathon apps
→ 0		
jeffdecola/resource-marathon-deploy		A Concourse resource type that deploys an
→ AP... 0		
ryanmehta/marathon-resource		
→ 0		
praekeltfoundation/marathon-acme		Automatically manage ACME certificates for
→ a... 0		[OK]
ckaznocha/marathon-resource		A Concourse resource to deploy applications
→ ... 0		

(continues on next page)

(continued from previous page)

quintoandar/drone-marathon ↳ 0	Drone plugin to create marathon deployments [OK]	↳
jamiecressey89/marathon-zookeeper ↳ for... 0	Zookeeper image that uses Marathon's API [OK]	↳
alenkacz/marathon-rabbitmq-autoscale ↳ in... 0	Autoscaling capabilities for apps running [OK]	↳
mrbobbytables/marathon ↳ 0	Marathon Mesos Framework container. [OK]	↳

- “Stars” indicate the popularity of the image.
- “Official” images are those in the root namespace.
- “Automated” images are built automatically by the Docker Hub.

(This means that their build recipe is always available.)

24.1.3.4.15 Downloading images

See also:

- <https://avril2018.container.training/intro.yml.html#143>

There are two ways to download images.

- Explicitly, with docker pull.
- Implicitly, when executing docker run and the image is not found locally.

24.1.3.4.16 Pulling an image

See also:

- <https://avril2018.container.training/intro.yml.html#144>

```
$ docker pull debian:jessie
```

```
jessie: Pulling from library/debian
3d77ce4481b1: Pull complete
Digest: sha256:f29d0c98d94d6b2169c740d498091a9a8545fabfa37f2072b43a4361c10064fc
Status: Downloaded newer image for debian:jessie
```

In this example, :jessie indicates which exact version of Debian we would like. **It is a version tag**.

24.1.3.4.17 Image and tags

See also:

- <https://avril2018.container.training/intro.yml.html#145>
- Images can have tags.
- Tags define image versions or variants.
- docker pull ubuntu will refer to ubuntu:latest.
- The :latest tag is generally updated often.

24.1.3.4.18 When to (not) use tags

See also:

- <https://avril2018.container.training/intro.yml.html#146>

Don't specify tags

- When doing rapid testing and prototyping.
- When experimenting.
- When you want the latest version.

Do specify tags

- When recording a procedure into a script.
- When going to production.
- To ensure that the same version will be used everywhere.
- To ensure repeatability later.

24.1.3.4.19 Section summary

See also:

- <https://avril2018.container.training/intro.yml.html#147>

We've learned how to:

- Understand images and layers.
- Understand Docker image namespacing.
- Search and download images.

24.1.4 Chapter3 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#9>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.4.1 Building images interactively

See also:

- <https://avril2018.container.training/intro.yml.html#149>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Building images interactively*
 - *Building images interactively*
 - *The plan*
 - *Setting up our container*
 - *Inspect the changes*
 - *Docker tracks filesystem changes*
 - *Copy-on-write security benefits*
 - *Commit our changes into a new image*
 - *Testing our new image*
 - *Tagging images*
 - *What's next ?*

24.1.4.1.1 Building images interactively

See also:

- <https://avril2018.container.training/intro.yml.html#150>

In this section, we will create our first container image.

It will be a basic distribution image, but we will pre-install the package figlet.

We will:

- Create a container from a base image.
- Install software manually in the container, and turn it into a new image.
- Learn about new commands:
 - *docker commit*,
 - *docker tag*,
 - and *docker diff*.

24.1.4.1.2 The plan

See also:

- <https://avril2018.container.training/intro.yml.html#151>
- Create a container (with docker run) using our base distro of choice.
- Run a bunch of commands to install and set up our software in the container.
- (Optionally) review changes in the container with docker diff.
- Turn the container into a new image with docker commit.
- (Optionally) add tags to the image with docker tag.

24.1.4.1.3 Setting up our container

See also:

- <https://avril2018.container.training/intro.yml.html#152>

Start an Ubuntu container:

```
$ docker run -it ubuntu
```

```
root@5d5da832b81a:/#
```

```
root@<yourContainerId>:#/
```

```
root@5d5da832b81a:/# apt-get update
```

```
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [83.2 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [65.5 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/universe Sources [11.5 MB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/universe Sources [3786 B]
Get:7 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [88.6 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [18.8 kB]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [1066 B]
Get:10 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/universe Sources [28.7 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [79.3 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [1660 B]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [127 kB]
Fetched 25.2 MB in 17s (1527 kB/s)
Reading package lists... Done
```

```
root@5d5da832b81a:/# apt-get install figlet
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 11 not upgraded.
Need to get 133 kB of archives.
After this operation, 752 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 figlet amd64 2.2.5-3 [133 kB]
Fetched 133 kB in 0s (382 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package figlet.
(Reading database ... 4035 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-3_amd64.deb ...
```

(continues on next page)

(continued from previous page)

```
Unpacking figlet (2.2.5-3) ...
Setting up figlet (2.2.5-3) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) ↵
↳ in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man6/figlet.6.gz ↵
↳ because associated file /usr/share/man/man6/figlet-figlet.6.gz (of link group ↵
↳ figlet) doesn't exist
```

24.1.4.1.4 Inspect the changes

See also:

- <https://avril2018.container.training/intro.yml.html#153>
- Open a new session into the docker server
- type docker ps to get the container id

```
[root@intranet-dev ~]# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	UP
STATUS	PORTS	NAMES		
814973c3cf61	ubuntu	" /bin/bash "	About a minute ago	Up
↳ About a minute		sharp_minsky		

Now let's run docker diff to see the difference between the base image and our container.

```
[root@intranet-dev ~]# docker diff 814
```

```
C /usr
C /usr/share
A /usr/share/figlet
A /usr/share/figlet/646-cn.flc
A /usr/share/figlet/646-de.flc
A /usr/share/figlet/646-gb.flc
A /usr/share/figlet/646-pt.flc
A /usr/share/figlet/8859-3.flc
A /usr/share/figlet/8859-4.flc
A /usr/share/figlet/koi8r.flc
A /usr/share/figlet/script.flf
A /usr/share/figlet/ushebrew.flc
A /usr/share/figlet/646-es2.flc
A /usr/share/figlet/646-hu.flc
A /usr/share/figlet/646-no.flc
A /usr/share/figlet/646-yu.flc
A /usr/share/figlet/ivrit.flf
A /usr/share/figlet/646-irv.flc
A /usr/share/figlet/frango.flc
```

24.1.4.1.5 Docker tracks filesystem changes

See also:

- <https://avril2018.container.training/intro.yml.html#154>

As explained before:

- An image is read-only.
- When we make changes, they happen in a copy of the image.
- Docker can show the difference between the image, and its copy.

For performance, Docker uses copy-on-write systems. (i.e. starting a container based on a big image doesn't incur a huge copy.)

24.1.4.1.6 Copy-on-write security benefits

See also:

- <https://avril2018.container.training/intro.yml.html#155>
- docker diff gives us an easy way to audit changes (à la Tripwire)
- Containers can also be started in read-only mode (their root filesystem will be read-only, but they can still have read-write data volumes)

24.1.4.1.7 Commit our changes into a new image

See also:

- <https://avril2018.container.training/intro.yml.html#156>

The docker commit command will create a new layer with those changes, and a new image using this new layer.

```
$ docker commit 814
```

```
sha256:c10a9dbc718b49ba25af4fc99d57c0fddd1dc87d3ab8f878caaeb135b4521f
```

The output of the docker commit command will be the ID for your newly created image.

We can use it as an argument to docker run.

24.1.4.1.8 Testing our new image

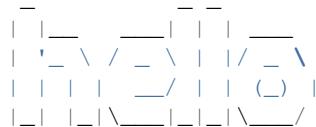
See also:

- <https://avril2018.container.training/intro.yml.html#157>

Let's run this image:

```
$ docker run -it c10a
```

```
root@d73eb40949d3:/# figlet hello
```



24.1.4.1.9 Tagging images

See also:

- <https://avril2018.container.training/intro.yml.html#158>

Referring to an image by its ID is not convenient. Let's tag it instead.

We can use the tag command:

```
$ docker tag c10a figlet
```

But we can also specify the tag as an extra argument to commit:

```
$ docker commit c10a figlet
```

And then run it using its tag::

```
$ docker run -it figlet
```

24.1.4.1.10 What's next ?

See also:

- <https://avril2018.container.training/intro.yml.html#159>

Manual process = bad. Automated process = good.

In the next chapter, we will learn how to automate the build process by writing a Dockerfile.

24.1.4.2 Building Docker images with a Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#161>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Building Docker images with a Dockerfile*
 - *Objectives*
 - *Dockerfile overview*
 - *Writing our first Dockerfile*
 - *Type this into our Dockerfile...*
 - *Build it!*
 - *What happens when we build the image ?*
 - *Sending the build context to Docker*
 - *Executing each step*
 - *The caching system*

- *Running the image*
- *Using image and viewing history*
- *Introducing JSON syntax*
- *JSON syntax vs string syntax*
- *When to use JSON syntax and string syntax*
 - * *String syntax*
 - * *JSON syntax*

24.1.4.2.1 Objectives

See also:

- <https://avril2018.container.training/intro.yml.html#163>

We will build a container image automatically, with a Dockerfile.

At the end of this lesson, you will be able to:

- Write a Dockerfile.
- Build an image from a Dockerfile.

24.1.4.2.2 Dockerfile overview

See also:

- <https://avril2018.container.training/intro.yml.html#164>
- A Dockerfile is a build recipe for a Docker image.
- It contains a series of instructions telling Docker how an image is constructed.
- The *docker build command* builds an image from a Dockerfile.

24.1.4.2.3 Writing our first Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#165>

Our Dockerfile must be in a new, empty directory.

1. Create a directory to hold our Dockerfile.
 - \$ mkdir myimage
2. Create a Dockerfile inside this directory.
 - \$ cd myimage
 - \$ vim Dockerfile

Of course, you can use any other editor of your choice.

24.1.4.2.4 Type this into our Dockerfile...

See also:

- <https://avril2018.container.training/intro.yml.html#166>

```
FROM ubuntu
RUN apt-get update
RUN apt-get install figlet
```

- FROM indicates the base image for our build.
- Each RUN line will be executed by Docker during the build.
- Our RUN commands must be non-interactive. (No input can be provided to Docker during the build.)

In many cases, we will add the -y flag to apt-get.

24.1.4.2.5 Build it!

See also:

- <https://avril2018.container.training/intro.yml.html#167>

Save our file, then execute:

```
$ docker build -t figlet .
```

- -t indicates the tag to apply to the image.
- . indicates the location of the build context.

We will talk more about the build context later.

To keep things simple for now: this is the directory where our Dockerfile is located.

24.1.4.2.6 What happens when we build the image ?

The output of docker build looks like this:

```
# docker build -t figlet .
```

```
Sending build context to Docker daemon 2.048kB
Step 1/3 : FROM ubuntu
latest: Pulling from library/ubuntu
a48c500ed24e: Already exists
1e1de00ff7e1: Already exists
0330ca45a200: Already exists
471db38bcfbf: Already exists
0b4aba487617: Already exists
Digest: sha256:c8c275751219dadad8fa56b3ac41ca6cb22219ff117ca98fe82b42f24e1ba64e
Status: Downloaded newer image for ubuntu:latest
--> 452a96d81c30
Step 2/3 : RUN apt-get update
--> Running in 81dab184c747
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [83.2 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [65.5 kB]
```

(continues on next page)

(continued from previous page)

```

Get:4 http://archive.ubuntu.com/ubuntu bionic/universe Sources [11.5 MB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security InRelease [83.2 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/universe Sources [3786 B]
Get:7 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 Packages [18.8 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [88.6 kB]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [1066 B]
Get:10 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/universe Sources [28.7 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [127 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 Packages [1660 B]
Get:17 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 Packages [79.3 kB]
Fetched 25.2 MB in 17s (1496 kB/s)
Reading package lists...
Removing intermediate container 81dab184c747
--> 01e04143b340
Step 3/3 : RUN apt-get install figlet
--> Running in 2dea10299bd1
Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
  figlet
0 upgraded, 1 newly installed, 0 to remove and 11 not upgraded.
Need to get 133 kB of archives.
After this operation, 752 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 figlet amd64 2.2.5-3 [133 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 133 kB in 22s (6148 B/s)
Selecting previously unselected package figlet.
(Reading database ... 4035 files and directories currently installed.)
Preparing to unpack .../figlet_2.2.5-3_amd64.deb ...
Unpacking figlet (2.2.5-3) ...
Setting up figlet (2.2.5-3) ...
update-alternatives: using /usr/bin/figlet-figlet to provide /usr/bin/figlet (figlet) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man6/figlet.6.gz because associated file /usr/share/man/man6/figlet-figlet.6.gz (of link group figlet) doesn't exist
Removing intermediate container 2dea10299bd1
--> e8fd21b0252b
Successfully built e8fd21b0252b
Successfully tagged figlet:latest

```

24.1.4.2.7 Sending the build context to Docker

See also:

- <https://avril2018.container.training/intro.yml.html#169>

Sending build context to Docker daemon 2.048 kB

- The build context is the . directory given to `docker build`
- It is sent (as an archive) by the Docker client to the Docker daemon.
- This allows to use a remote machine to build using local files.
- Be careful (or patient) if that directory is big and your link is slow.

24.1.4.2.8 Executing each step

See also:

- <https://avril2018.container.training/intro.yml.html#169>

24.1.4.2.9 The caching system

See also:

- <https://avril2018.container.training/intro.yml.html#171>

If you run the same build again, it will be instantaneous. Why ?

After each build step, Docker takes a snapshot of the resulting image.

Before executing a step, Docker checks if it has already built the same sequence.

Docker uses the exact strings defined in your Dockerfile, so:

- RUN apt-get install figlet cowsay is different from RUN apt-get install cowsay figlet
- RUN apt-get update is not re-executed when the mirrors are updated

You can force a rebuild with docker build –no-cache

```
# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
figlet	latest	e8fd21b0252b	10 minutes ago	121MB
ubuntu	latest	452a96d81c30	4 weeks ago	79.6MB

24.1.4.2.10 Running the image

See also:

- <https://avril2018.container.training/intro.yml.html#172>

The resulting image is not different from the one produced manually.

24.1.4.2.11 Using image and viewing history

See also:

- <https://avril2018.container.training/intro.yml.html#173>

The `history` command lists all the layers composing an image.

For each layer, it shows its creation time, size, and creation command.

When an image was built with a Dockerfile, each layer corresponds to a line of the Dockerfile.

24.1.4.2.12 Introducing JSON syntax

See also:

- <https://avril2018.container.training/intro.yml.html#174>

Most Dockerfile arguments can be passed in two forms:

- plain string: RUN apt-get install figlet
- JSON list: RUN ["apt-get", "install", "figlet"]

We are going to change our Dockerfile to see how it affects the resulting image.

24.1.4.2.13 JSON syntax vs string syntax

See also:

- <https://avril2018.container.training/intro.yml.html#176>

Compare the new history:

IMAGE	CREATED	CREATED BY
	SIZE	COMMENT
ba8d944addee0	39 seconds ago	apt-get install figlet
↪ 992kB		
01e04143b340	18 minutes ago	/bin/sh -c apt-get update
↪ 40.5MB		
452a96d81c30	4 weeks ago	/bin/sh -c #(nop) CMD ["/bin/bash"]
↪ 0B		
<missing>	4 weeks ago	/bin/sh -c mkdir -p /run/systemd && echo
↪ 'do... 7B		
<missing>	4 weeks ago	/bin/sh -c sed -i 's/^#\s*\(\deb.*universe\)\s*/' /etc/apt/sources.list
↪ \$... 2.76kB		
<missing>	4 weeks ago	/bin/sh -c rm -rf /var/lib/apt/lists/*
↪ 0B		
<missing>	4 weeks ago	/bin/sh -c set -xe && echo '#!/bin/sh' > /etc/rc.local
↪ ... 745B		
<missing>	4 weeks ago	/bin/sh -c #(nop) ADD
↪ file:81813d6023adb66b8...	79.6MB	

- JSON syntax specifies an exact command to execute.
- String syntax specifies a command to be wrapped within /bin/sh -c “...”.

24.1.4.2.14 When to use JSON syntax and string syntax

See also:

- <https://avril2018.container.training/intro.yml.html#177>

String syntax

- is easier to write
- interpolates environment variables and other shell expressions
- creates an extra process (`/bin/sh -c ...`) to parse the string
- requires `/bin/sh` to exist in the container

JSON syntax

- is harder to write (and read!)
- passes all arguments without extra processing
- doesn't create an extra process
- doesn't require `/bin/sh` to exist in the container

24.1.4.3 CMD and ENTRYPOINT

See also:

- <https://avril2018.container.training/intro.yml.html#179>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *CMD and ENTRYPOINT*
 - *Objectives*
 - *Defining a default command*
 - *Adding CMD to our Dockerfile*
 - *Build and test our image*
 - *Overriding CMD*
 - *Using ENTRYPOINT*
 - *Adding ENTRYPOINT to our Dockerfile*
 - *Implications of JSON vs string syntax*
 - *Build and test our image*
 - *Using CMD and ENTRYPOINT together*
 - *CMD and ENTRYPOINT together*
 - *Build and test our image*
 - *Overriding the image default parameters*
 - *Overriding ENTRYPOINT*

24.1.4.3.1 Objectives

See also:

- <https://avril2018.container.training/intro.yml.html#181>

In this lesson, we will learn about two important Dockerfile commands:

- CMD and ENTRYPOINT.

These commands allow us to set the default command to run in a container.

24.1.4.3.2 Defining a default command

See also:

- <https://avril2018.container.training/intro.yml.html#182>

When people run our container, we want to greet them with a nice hello message, and using a custom font.

For that, we will execute:

```
figlet -f script hello
```

- -f script tells figlet to use a fancy font.
- hello is the message that we want it to display.

24.1.4.3.3 Adding CMD to our Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#183>

Our new Dockerfile will look like this:

```
FROM ubuntu
RUN apt-get update
RUN ["apt-get", "install", "figlet"]
CMD figlet -f script hello
```

- CMD defines a default command to run when none is given.
- It can appear at any point in the file.
- Each CMD will replace and override the previous one.
- As a result, **while you can have multiple CMD lines, it is useless.**

24.1.4.3.4 Build and test our image

See also:

- <https://avril2018.container.training/intro.yml.html#184>

```
docker build -t figlet .
```

```

Sending build context to Docker daemon 2.048kB
Step 1/4 : FROM ubuntu
--> 452a96d81c30
Step 2/4 : RUN apt-get update
--> Using cache
--> 01e04143b340
Step 3/4 : RUN ["apt-get", "install", "figlet"]
--> Using cache
--> ba8d944addee0
Step 4/4 : CMD figlet -f script hello
--> Running in 9ab6d5965c4c
Removing intermediate container 9ab6d5965c4c
--> d06ea4383cc6
Successfully built d06ea4383cc6
Successfully tagged figlet:latest

```

```
# docker run figlet
```

24.1.4.3.5 Overriding CMD

See also:

- <https://avril2018.container.training/intro.yml.html#185>

If we want to get a shell into our container (instead of running figlet), we just have to specify a different program to run:

```
$ docker run -it figlet bash
```

- We specified bash.
- It replaced the value of CMD.

24.1.4.3.6 Using ENTRYPPOINT

See also:

- <https://avril2018.container.training/intro.yml.html#186>

We want to be able to specify a different message on the command line, while retaining figlet and some default parameters.

In other words, we would like to be able to do this:

```
$ docker run figlet salut
```

We will use the ENTRYPPOINT verb in Dockerfile.

24.1.4.3.7 Adding ENTRYPPOINT to our Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#187>

Our new Dockerfile will look like this:

```
FROM ubuntu
RUN apt-get update
RUN ["apt-get", "install", "figlet"]
ENTRYPOINT ["figlet", "-f", "script"]
```

- ENTRYPOINT defines a base command (and its parameters) for the container.
- The command line arguments are appended to those parameters.
- Like CMD, ENTRYPOINT can appear anywhere, and replaces the previous value.

Why did we use JSON syntax for our ENTRYPOINT ?

24.1.4.3.8 Implications of JSON vs string syntax

See also:

- <https://avril2018.container.training/intro.yml.html#188>
- When CMD or ENTRYPOINT use string syntax, they get wrapped in sh -c.
- To avoid this wrapping, we can use JSON syntax.

What if we used ENTRYPOINT with string syntax ?

```
$ docker run figlet salut
```

This would run the following command in the figlet image:

```
sh -c "figlet -f script" salut
```

24.1.4.3.9 Build and test our image

See also:

- <https://avril2018.container.training/intro.yml.html#189>

Let's build it:

```
$ docker build -t figlet .
Successfully built cede00171081
Successfully tagged figlet:latest
```

And run it:

```
$ docker run figlet salut
```

24.1.4.3.10 Using CMD and ENTRYPOINT together

See also:

- <https://avril2018.container.training/intro.yml.html#190>

What if we want to define a default message for our container?

Then we will use ENTRYPOINT and CMD together.

- ENTRYPPOINT will define the base command for our container.
- CMD will define the default parameter(s) for this command.

They both have to use JSON syntax.

24.1.4.3.11 CMD and ENTRYPPOINT together

See also:

- <https://avril2018.container.training/intro.yml.html#191>

Our new Dockerfile will look like this:

```
FROM ubuntu
RUN apt-get update
RUN ["apt-get", "install", "figlet"]
ENTRYPOINT ["figlet", "-f", "script"]
CMD ["hello world"]
```

- ENTRYPPOINT defines a base command (and its parameters) for the container.
- If we don't specify extra command-line arguments when starting the container, the value of CMD is appended.
- Otherwise, our extra command-line arguments are used instead of CMD.

24.1.4.3.12 Build and test our image

See also:

- <https://avril2018.container.training/intro.yml.html#192>

Let's build it:

```
# docker build -t figlet .
```

```
Sending build context to Docker daemon 2.048kB
Step 1/5 : FROM ubuntu
--> 452a96d81c30
Step 2/5 : RUN apt-get update
--> Using cache
--> 01e04143b340
Step 3/5 : RUN ["apt-get", "install", "figlet"]
--> Using cache
--> ba8d944addee
Step 4/5 : ENTRYPOINT ["figlet", "-f", "script"]
--> Using cache
--> cede00171081
Step 5/5 : CMD ["hello world"]
--> Running in 5019ef053005
Removing intermediate container 5019ef053005
--> 848a294a8347
Successfully built 848a294a8347
Successfully tagged figlet:latest
```

24.1.4.3.13 Overriding the image default parameters

See also:

- <https://avril2018.container.training/intro.yml.html#193>

Now let's pass extra arguments to the image.

```
docker run figlet hola mundo
```



We overrode CMD but still used ENTRYPOINT.

24.1.4.3.14 Overriding ENTRYPOINT

See also:

- <https://avril2018.container.training/intro.yml.html#194>

What if we want to run a shell in our container ?

We cannot just do docker run figlet bash because that would just tell figlet to display the word "bash."

We use the --entrypoint parameter:

```
$ docker run -it --entrypoint bash figlet
```

24.1.4.4 Copying files during the build

See also:

- <https://avril2018.container.training/intro.yml.html#196>
- <https://avril2018.container.training/intro.yml.html#9>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Copying files during the build*
 - *Objectives*
 - *Build some C code*
 - *The Dockerfile*
 - *Testing our C program*
 - *COPY and the build cache*
 - *Details*
 - *Next step : multi-stage building*

24.1.4.4.1 Objectives

See also:

- <https://avril2018.container.training/intro.yml.html#198>

So far, we have installed things in our container images by downloading packages.

We can also copy files from the **build context** to the container that we are building.

Remember: the build context is the directory containing the Dockerfile.

In this chapter, we will learn a new Dockerfile keyword: COPY.

24.1.4.4.2 Build some C code

See also:

- <https://avril2018.container.training/intro.yml.html#199>

We want to build a container that compiles a basic “Hello world” program in C.

Here is the program, hello.c:

```
int main () {
    puts("Hello, world!");
    return 0;
}
```

Let's create a new directory, and put this file in there.

Then we will write the Dockerfile.

24.1.4.4.3 The Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#200>

On Debian and Ubuntu, the package build-essential will get us a compiler.

When installing it, don't forget to specify the -y flag, otherwise the build will fail (since the build cannot be interactive).

Then we will use COPY to place the source file into the container.

```
FROM ubuntu
RUN apt-get update
RUN apt-get install -y build-essential
COPY hello.c /
RUN make hello
CMD /hello
```

Create this Dockerfile.

24.1.4.4.4 Testing our C program

See also:

- <https://avril2018.container.training/intro.yml.html#201>

- Create hello.c and Dockerfile in the same directory.
- Run docker build -t hello . in this directory.
- Run docker run hello, you should see Hello, world!.

```
# docker run hello
```

```
Hello, world!
```

24.1.4.4.5 COPY and the build cache

See also:

- <https://avril2018.container.training/intro.yml.html#202>
- Run the build again.
- Now, modify hello.c and run the build again.
- Docker can cache steps involving COPY.
- Those steps will not be executed again if the files haven't been changed.

```
[root@intranet-dev myc]# docker build -t hello .
```

```
Sending build context to Docker daemon 3.072kB
Step 1/6 : FROM ubuntu
--> 452a96d81c30
Step 2/6 : RUN apt-get update
--> Using cache
--> 01e04143b340
Step 3/6 : RUN apt-get install -y build-essential
--> Using cache
--> 9139dae8927e
Step 4/6 : COPY hello.c /
--> c803db9440ed
Step 5/6 : RUN make hello
--> Running in 3f92e8e74085
cc      hello.c  -o hello
hello.c: In function 'main':
hello.c:2:3: warning: implicit declaration of function 'puts' [-Wimplicit-function-
declaration]
    puts("Hello, big world!");
    ^
~~~
Removing intermediate container 3f92e8e74085
--> 2d25a58a49f0
Step 6/6 : CMD /hello
--> Running in be79b29a07e0
Removing intermediate container be79b29a07e0
--> aae25a3dfa28
Successfully built aae25a3dfa28
Successfully tagged hello:latest
```

24.1.4.6 Details

See also:

- <https://avril2018.container.training/intro.yml.html#203>
- You can COPY whole directories recursively.
- Older Dockerfiles also have the ADD instruction. It is similar but can automatically extract archives.
- If we really wanted to compile C code in a container, we would:
 - Place it in a different directory, with the WORKDIR instruction.
 - Even better, use the gcc official image.

24.1.4.4.7 Next step : multi-stage building

- *Multi-stage builds*

24.1.4.5 Multi-stage builds

See also:

- <https://avril2018.container.training/intro.yml.html#205>
- <https://avril2018.container.training/intro.yml.html#9>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Multi-stage builds*
 - *Multi-stage builds*
 - *Multi-stage builds principles*
 - *Multi-stage builds in practice*
 - *Multi-stage builds for our C program*
 - *Multi-stage build Dockerfile*
 - * *Before the build*
 - *Comparing single/multi-stage build image sizes*

24.1.4.5.1 Multi-stage builds

See also:

- <https://avril2018.container.training/intro.yml.html#206>

In the previous example, our final image contain:

- our hello program
- its source code
- the compiler

Only the first one is strictly necessary.

We are going to see how to obtain an image without the superfluous components.

24.1.4.5.2 Multi-stage builds principles

See also:

- <https://avril2018.container.training/intro.yml.html#207>
- At any point in our Dockerfile, we can add a new FROM line.
- This line starts a new stage of our build.
- Each stage can access the files of the previous stages with COPY --from=....
- When a build is tagged (with docker build -t ...), the last stage is tagged.
- Previous stages are not discarded: they will be used for caching, and can be referenced.

24.1.4.5.3 Multi-stage builds in practice

See also:

- <https://avril2018.container.training/intro.yml.html#208>

Each stage is numbered, starting at 0

We can copy a file from a previous stage by indicating its number, e.g.

```
COPY --from=0 /file/from/first/stage /location/in/current/stage
```

We can also name stages, and reference these names

```
FROM golang AS builder
RUN ...
FROM alpine
COPY --from=builder /go/bin/mylittlebinary /usr/local/bin/
```

24.1.4.5.4 Multi-stage builds for our C program

See also:

- <https://avril2018.container.training/intro.yml.html#209>

We will change our Dockerfile to:

- give a nickname to the first stage: compiler
- add a second stage using the same ubuntu base image
- add the hello binary to the second stage
- make sure that CMD is in the second stage

The resulting Dockerfile is on the next slide.

24.1.4.5.5 Multi-stage build Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#210>

Here is the final Dockerfile:

```
FROM ubuntu AS compiler
RUN apt-get update
RUN apt-get install -y build-essential
COPY hello.c /
RUN make hello
FROM ubuntu
COPY --from=compiler /hello /hello
CMD /hello
```

Let's build it, and check that it works correctly:

Before the build

```
# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello	latest	aae25a3dfa28	30 minutes ago	325MB
<none>	<none>	e43bb6363c1f	42 minutes ago	325MB
ubuntu	latest	452a96d81c30	4 weeks ago	79.6MB

```
# docker build -t hellomultistage .
```

```
Sending build context to Docker daemon 3.072kB
Step 1/8 : FROM ubuntu AS compiler
--> 452a96d81c30
Step 2/8 : RUN apt-get update
--> Using cache
--> 01e04143b340
Step 3/8 : RUN apt-get install -y build-essential
--> Using cache
--> 9139dae8927e
Step 4/8 : COPY hello.c /
--> Using cache
--> c803db9440ed
Step 5/8 : RUN make hello
--> Using cache
--> 2d25a58a49f0
Step 6/8 : FROM ubuntu
--> 452a96d81c30
Step 7/8 : COPY --from=compiler /hello /hello
--> d427a7aa53af
Step 8/8 : CMD /hello
--> Running in f338055a571e
Removing intermediate container f338055a571e
--> c8be88f00576
Successfully built c8be88f00576
Successfully tagged hellomultistage:latest
```

```
# docker run hellomultistage
```

```
Hello, big world!
```

24.1.4.5.6 Comparing single/multi-stage build image sizes

See also:

- <https://avril2018.container.training/intro.yml.html#211>

List our images with docker images, and check the size of:

- the ubuntu base image (79.6MB)
- the single-stage hello image (325MB)
- the multi-stage hellomultistage image (79.6MB)

We can achieve even smaller images if we use smaller base images.

However, if we use common base images (e.g. if we standardize on ubuntu), these common images will be pulled only once per node, so they are virtually “free.”

docker images					
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE	
hellomultistage	latest	c8be88f00576	About a minute ago	79.	↪ 6MB
hello	latest	aae25a3dfa28	34 minutes ago	325MB	<none>
ubuntu	latest	e43bb6363c1f	About an hour ago	325MB	<none>
		452a96d81c30	4 weeks ago	79.	↪ 6MB

24.1.4.6 Publishing images to the Docker Hub

See also:

- <https://avril2018.container.training/intro.yml.html#213>
- <https://avril2018.container.training/intro.yml.html#9>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Publishing images to the Docker Hub*
 - *Publishing images to the Docker Hub*
 - *Logging into our Docker Hub account*
 - *Image tags and registry addresses*
 - *Image tags and registry addresses*
 - *Tagging an image to push it on the Hub*
 - * *figlet Dockerfile*

24.1.4.6.1 Publishing images to the Docker Hub

See also:

- <https://avril2018.container.training/intro.yml.html#214>

We have built our first images.

We can now publish it to the Docker Hub!

You don't have to do the exercises in this section, because they require an account on the Docker Hub, and we don't want to force anyone to create one.

Note, however, that creating an account on the Docker Hub is free (and doesn't require a credit card), and hosting public images is free as well.

24.1.4.6.2 Logging into our Docker Hub account

See also:

- <https://avril2018.container.training/intro.yml.html#215>

This can be done from the Docker CLI:

```
docker login
```

```
Login with your Docker ID to push and pull images from Docker Hub.  
If you don't have a Docker ID, head over to https://hub.docker.com  
to create one.
```

```
Username: id3pvergain
```

```
Password:
```

```
Login Succeeded
```

24.1.4.6.3 Image tags and registry addresses

See also:

- <https://avril2018.container.training/intro.yml.html#216>
- Docker images tags are like Git tags and branches.
- They are like bookmarks pointing at a specific image ID.
- Tagging an image doesn't rename an image: it adds another tag.
- When pushing an image to a registry, the registry address is in the tag.
- Example: `registry.example.net:5000/image`
- What about Docker Hub images?

24.1.4.6.4 Image tags and registry addresses

See also:

- <https://avril2018.container.training/intro.yml.html#217>

- Docker images tags are like Git tags **and** branches.
- They are like bookmarks pointing at a specific image ID.
- Tagging an image doesn't rename an image: it adds another tag.
- When pushing an image to a registry, the registry address **is in** the tag.
- Example: `registry.example.net:5000/image`

- What about Docker Hub images?
- `jpetazzo/clock` is, in fact, `index.docker.io/jpetazzo/clock`
- `ubuntu` is, in fact, `library/ubuntu`, i.e. `index.docker.io/library/ubuntu`

24.1.4.6.5 Tagging an image to push it on the Hub

See also:

- <https://avril2018.container.training/intro.yml.html#218>
- <https://avril2018.container.training/intro.yml.html#219>

figlet Dockerfile

```
# cat Dockerfile
```

```
FROM ubuntu
RUN apt-get update
RUN ["apt-get", "install", "figlet"]
ENTRYPOINT ["figlet", "-f", "script"]
CMD ["hello world"]
```

Let's tag our figlet image (or any other to our liking)

```
docker tag figlet id3pvergain/figlet
```

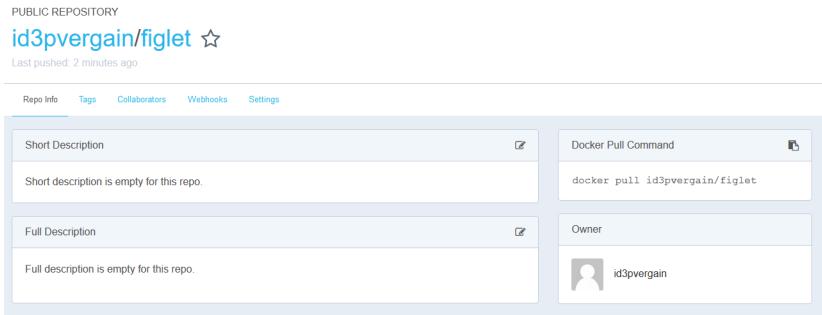
```
docker push id3pvergain/figlet
```

```
[root@intranet-dev myimage]# docker tag figlet id3pvergain/figlet
[root@intranet-dev myimage]# docker push id3pvergain/figlet
The push refers to repository [docker.io/id3pvergain/figlet]
6a460659e0ae: Pushed
3023de562a6f: Pushed
059ad60bcacf: Mounted from library/ubuntu
8db5f072feec: Mounted from library/ubuntu
67885e448177: Mounted from library/ubuntu
ec75999a0cb1: Mounted from library/ubuntu
65bdd50ee76a: Mounted from library/ubuntu
latest: digest: ↵
sha256:b239196e33c151a85c6bea76bb3eeecadea8ea43d811d0d3aba7ed32efa9e919 size: 1779
```

Anybody can now docker run `id3pvergain/figlet` anywhere.

24.1.4.7 Tips for efficient Dockerfiles

See also:



- <https://avril2018.container.training/intro.yml.html#223>
- <https://avril2018.container.training/intro.yml.html#9>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Tips for efficient Dockerfiles*
 - *Tips for efficient Dockerfiles*
 - *Reducing the number of layers*
 - *Avoid re-installing dependencies at each build*
 - *Example “bad” Dockerfile*
 - *Fixed Dockerfile*
 - *Embedding unit tests in the build process*

24.1.4.7.1 Tips for efficient Dockerfiles

See also:

- <https://avril2018.container.training/intro.yml.html#224>

We will see how to:

- Reduce the number of layers.
- Leverage the build cache so that builds can be faster.
- Embed unit testing in the build process.

24.1.4.7.2 Reducing the number of layers

See also:

- <https://avril2018.container.training/intro.yml.html#225>
- Each line in a Dockerfile creates a new layer.
- Build your Dockerfile to take advantage of Docker’s caching system.
- Combine commands by using && to continue commands and to wrap lines.

Note: it is frequent to build a Dockerfile line by line

```
RUN apt-get install thisthing  
RUN apt-get install andthatthing andthatotherone  
RUN apt-get install somemorestuff
```

And then refactor it trivially before shipping

```
RUN apt-get install thisthing andthatthing andthatotherone somemorestuff
```

24.1.4.7.3 Avoid re-installing dependencies at each build

See also:

- <https://avril2018.container.training/intro.yml.html#226>
- Classic Dockerfile problem: “each time I change a line of code, all my dependencies are re-installed!”
- Solution: COPY dependency lists (package.json, requirements.txt, etc.) by themselves to avoid reinstalling unchanged dependencies every time

24.1.4.7.4 Example “bad” Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#227>

The dependencies are reinstalled every time, because the build system does not know if requirements.txt has been updated.

```
FROM python  
MAINTAINER Docker Education Team <education@docker.com>  
COPY . /src/  
WORKDIR /src  
RUN pip install -qr requirements.txt  
EXPOSE 5000  
CMD ["python", "app.py"]
```

24.1.4.7.5 Fixed Dockerfile

See also:

- <https://avril2018.container.training/intro.yml.html#228>
- <https://docs.docker.com/engine/deprecated/#maintainer-in-dockerfile>

Note: MAINTAINER was an early very limited form of LABEL which should be used instead.

- <https://docs.docker.com/engine/deprecated/#maintainer-in-dockerfile>

The recommended solution is to use LABEL instead, e.g. LABEL authors=”first author,second author”

Adding the dependencies as a separate step means that Docker can cache more efficiently and only install them when requirements.txt changes.

```
FROM python
MAINTAINER Docker Education Team <education@docker.com>
COPY ./requirements.txt /tmp/requirements.txt
RUN pip install -qr /tmp/requirements.txt
COPY . /src/
WORKDIR /src
EXPOSE 5000
CMD ["python", "app.py"]
```

24.1.4.7.6 Embedding unit tests in the build process

See also:

- <https://avril2018.container.training/intro.yml.html#229>

```
FROM <baseimage>
RUN <install dependencies>
COPY <code>
RUN <build code>
RUN <install test dependencies>
COPY <test data sets and fixtures>
RUN <unit tests>
FROM <baseimage>
RUN <install dependencies>
COPY <code>
RUN <build code>
CMD, EXPOSE ...
```

- The build fails as soon as an instruction fails
- If RUN <unit tests> fails, the build doesn't produce an image
- If it succeeds, it produces a clean image (without test libraries and data)

24.1.5 Chapter4 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#231>
- <https://avril2018.container.training/intro.yml.html#9>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.5.1 Naming and inspecting containers

See also:

- <https://avril2018.container.training/intro.yml.html#231>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Naming and inspecting containers*
 - *Objectives*
 - *Naming our containers*
 - *Default names*
 - *Specifying a name*
 - *Renaming containers*
 - *Inspecting a container*
 - *Parsing JSON with the Shell*
 - *Using –format*

24.1.5.1.1 Objectives

See also:

- <https://avril2018.container.training/intro.yml.html#233>

In this lesson, we will learn about an important Docker concept: container naming.

Naming allows us to:

- Reference easily a container.
- Ensure unicity of a specific container.

We will also see the *inspect command*, which gives a lot of details about a container.

24.1.5.1.2 Naming our containers

See also:

- <https://avril2018.container.training/intro.yml.html#234>

So far, we have referenced containers with their ID.

We have copy-pasted the ID, or used a shortened prefix.

But each container can also be referenced by its name.

If a container is named thumbnail-worker, I can do:

```
$ docker logs thumbnail-worker
$ docker stop thumbnail-worker
etc.
```

24.1.5.1.3 Default names

See also:

- <https://avril2018.container.training/intro.yml.html#235>

When we create a container, if we don't give a specific name, Docker will pick one for us.

It will be the concatenation of:

- A mood (furious, goofy, suspicious, boring...)
- The name of a famous inventor (tesla, darwin, wozniak...)

Examples: happy_curie, clever_hopper, jovial_lovelace ...

24.1.5.1.4 Specifying a name

See also:

- <https://avril2018.container.training/intro.yml.html#236>

You can set the name of the container when you create it.

```
$ docker run --name ticktock jpetazzo/clock
```

If you specify a name that already exists, Docker will refuse to create the container.

This lets us enforce unicity of a given resource.

24.1.5.1.5 Renaming containers

See also:

- <https://avril2018.container.training/intro.yml.html#237>

You can rename containers with *docker rename*. This allows you to “free up” a name without destroying the associated container.

24.1.5.1.6 Inspecting a container

See also:

- <https://avril2018.container.training/intro.yml.html#238>

The docker inspect command will output a very detailed JSON map.

```
$ docker inspect <containerID>
```

```
[ {  
  ...  
(many pages of JSON here)  
  ...}
```

There are multiple ways to consume that information

24.1.5.1.7 Parsing JSON with the Shell

See also:

- <https://avril2018.container.training/intro.yml.html#239>
- You could grep and cut or awk the output of docker inspect.

- Please, don't.
- It's painful.
- If you really must parse JSON from the Shell, use JQ! (It's great.)

```
$ docker inspect <containerID> | jq .
```

We will see a better solution which doesn't require extra tools.

24.1.5.1.8 Using –format

See also:

- <https://avril2018.container.training/intro.yml.html#240>

You can specify a format string, which will be parsed by Go's text/template package.

```
$ docker inspect --format '{{ json .Created }}' <containerID>
"2015-02-24T07:21:11.712240394Z"
```

- The generic syntax is to wrap the expression with double curly braces.
- The expression starts with a dot representing the JSON object.
- Then each field or member can be accessed in dotted notation syntax.
- The optional json keyword asks for valid JSON output. (e.g. here it adds the surrounding double-quotes.)

24.1.5.2 Naming and inspecting containers

See also:

- <https://avril2018.container.training/intro.yml.html#242>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Naming and inspecting containers*
 - *Labels*
 - *Using labels*
 - *Querying labels*
 - *Using labels to select containers (docker ps –filter)*
 - *Use-cases for labels*

24.1.5.2.1 Labels

See also:

- <https://avril2018.container.training/intro.yml.html#243>

- Labels allow to attach arbitrary metadata to containers.
- Labels are key/value pairs.
- They are specified at container creation
- You can query them with docker inspect.
- They can also be used as filters with some commands (e.g. docker ps).

24.1.5.2.2 Using labels

See also:

- <https://avril2018.container.training/intro.yml.html#244>

Let's create a few containers with a label owner:

```
docker run -d -l owner=alice nginx
docker run -d -l owner=bob nginx
docker run -d -l owner nginx
```

We didn't specify a value for the owner label in the last example.

This is equivalent to setting the value to be an empty string.

24.1.5.2.3 Querying labels

See also:

- <https://avril2018.container.training/intro.yml.html#245>

We can view the labels with docker inspect.

```
$ docker inspect $(docker ps -lq) | grep -A3 Labels
    "Labels": {
        "maintainer": "NGINX Docker Maintainers <docker-
        ↪maint@nginx.com>",
        "owner": ""
    },
```

We can use the --format flag to list the value of a label.

```
$ docker inspect $(docker ps -q) --format 'OWNER={{.Config.Labels.owner}}'
```

24.1.5.2.4 Using labels to select containers (docker ps --filter)

See also:

- <https://avril2018.container.training/intro.yml.html#246>

We can list containers having a specific label.

```
$ docker ps --filter label=owner
```

Or we can list containers having a specific label with a specific value.

```
$ docker ps --filter label=owner=alice
```

24.1.5.2.5 Use-cases for labels

See also:

- <https://avril2018.container.training/intro.yml.html#246>
- HTTP vhost of a web app or web service. (The label is used to generate the configuration for NGINX, HAProxy, etc.)
- Backup schedule for a stateful service. (The label is used by a cron job to determine if/when to backup container data.)
- Service ownership. (To determine internal cross-billing, or who to page in case of outage.)
- etc.

24.1.5.3 Getting inside a container

See also:

- <https://avril2018.container.training/intro.yml.html#249>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Getting inside a container*
 - *Objectives*
 - *Getting a shell*
 - *Not getting a shell*
 - *Viewing container processes from the host*
 - *What's the difference between a container process and a host process ?*
 - *Getting a shell in a running container*
 - *Caveats*
 - *Getting a shell in a stopped container*
 - *Analyzing a stopped container*
 - *Viewing filesystem changes*
 - *Accessing files*
 - *Exploring a crashed container (docker commit + docker run -ti --entrypoint)*
 - *Obtaining a complete dump (docker export)*

24.1.5.3.1 Objectives

See also:

- <https://avril2018.container.training/intro.yml.html#251>

On a traditional server or VM, we sometimes need to:

- log into the machine (with SSH or on the console),
- analyze the disks (by removing them or rebooting with a rescue system).

In this chapter, we will see how to do that with containers.

24.1.5.3.2 Getting a shell

See also:

- <https://avril2018.container.training/intro.yml.html#252>

Every once in a while, we want to log into a machine.

In a perfect world, this shouldn't be necessary.

- You need to install or update packages (and their configuration)?
- Use configuration management. (e.g. Ansible, Chef, Puppet, Salt...)
- You need to view logs and metrics?
- Collect and access them through a centralized platform.

In the real world, though ... we often need shell access!

24.1.5.3.3 Not getting a shell

See also:

- <https://avril2018.container.training/intro.yml.html#253>

Even without a perfect deployment system, we can do many operations without getting a shell.

- Installing packages can (and should) be done in the container image.
- Configuration can be done at the image level, or when the container starts.
- Dynamic configuration can be stored in a volume (shared with another container).
- Logs written to stdout are automatically collected by the Docker Engine.
- Other logs can be written to a shared volume.
- Process information and metrics are visible from the host.

Let's save logging, volumes ... for later, but let's have a look at process information !

24.1.5.3.4 Viewing container processes from the host

See also:

- <https://avril2018.container.training/intro.yml.html#254>

If you run Docker on Linux, container processes are visible on the host.

```
$ ps faux | less
```

- Scroll around the output of this command.
- You should see the jpetazzo/clock container.
- A containerized process is just like any other process on the host.
- We can use tools like lsof, strace, gdb ... To analyze them.

24.1.5.3.5 What's the difference between a container process and a host process ?

See also:

- <https://avril2018.container.training/intro.yml.html#255>
- Each process (containerized or not) belongs to namespaces and cgroups.
- The namespaces and cgroups determine what a process can “see” and “do”.
- Analogy: each process (containerized or not) runs with a specific UID (user ID).
- UID=0 is root, and has elevated privileges. Other UIDs are normal users.

We will give more details about namespaces and cgroups later.

24.1.5.3.6 Getting a shell in a running container

See also:

- <https://avril2018.container.training/intro.yml.html#256>
- Sometimes, we need to get a shell anyway.
- We could run some SSH server in the container ...
- But it is easier to use docker exec.

```
$ docker exec -ti ticktock sh
```

- This creates a new process (running sh) inside the container.
- This can also be done “manually” with the tool nsenter.

24.1.5.3.7 Caveats

See also:

- <https://avril2018.container.training/intro.yml.html#257>
- The tool that you want to run needs to exist in the container.
- Some tools (like ip netns exec) let you attach to one namespace at a time.
- (This lets you e.g. setup network interfaces, even if you don't have ifconfig or ip in the container.)
- Most importantly: the container needs to be running.
- What if the container is stopped or crashed?

24.1.5.3.8 Getting a shell in a stopped container

See also:

- <https://avril2018.container.training/intro.yml.html#258>
- A stopped container is only storage (like a disk drive).
- We cannot SSH into a disk drive or USB stick!
- We need to connect the disk to a running machine.
- How does that translate into the container world?

24.1.5.3.9 Analyzing a stopped container

See also:

- <https://avril2018.container.training/intro.yml.html#259>

As an exercise, we are going to try to find out what's wrong with jpetazzo/crashtest.

```
docker run jpetazzo/crashtest
```

The container starts, but then stops immediately, without any output.

What would McGyver do ?

First, let's check the status of that container.

```
docker ps -l
```

24.1.5.3.10 Viewing filesystem changes

See also:

- <https://avril2018.container.training/intro.yml.html#260>
- We can use docker diff to see files that were added / changed / removed.

`docker diff <container_id>`

- The container ID was shown by `docker ps -l`.
- We can also see it with `docker ps -lq`.
- The output of `docker diff` shows some interesting log files!

```
[root@intranet-dev projects]# docker diff 9a4
```

```
C /var
C /var/log
C /var/log/nginx
A /var/log/nginx/error.log
A /var/log/nginx/access.log
C /var/tmp
C /var/tmp/nginx
A /var/tmp/nginx/proxy
A /var/tmp/nginx/scgi
```

(continues on next page)

(continued from previous page)

```
A /var/tmp/nginx/uwsgi
A /var/tmp/nginx/client_body
A /var/tmp/nginx/fastcgi
```

24.1.5.3.11 Accessing files

See also:

- <https://avril2018.container.training/intro.yml.html#261>
- We can extract files with docker cp.

```
docker cp <container_id>:/var/log/nginx/error.log .
```

Then we can look at that log file.

```
cat error.log
```

!! ca ne marche pas pour moi.

24.1.5.3.12 Exploring a crashed container (docker commit + docker run -ti --entrypoint)

See also:

- <https://avril2018.container.training/intro.yml.html#262>
- We can restart a container with docker start ...
- ... But it will probably crash again immediately!
- We cannot specify a different program to run with docker start
- But we can create a new image from the crashed container

```
docker commit <container_id> debugimage
```

Then we can run a new container from that image, with a custom entrypoint

```
docker run -ti --entrypoint sh debugimage
```

24.1.5.3.13 Obtaining a complete dump (docker export)

See also:

- <https://avril2018.container.training/intro.yml.html#263>
- We can also dump the entire filesystem of a container.
- This is done with docker export.
- It generates a tar archive.

```
docker export <container_id> | tar tv
```

This will give a detailed listing of the content of the container.

24.1.5.4 Container networking basics

See also:

- <https://avril2018.container.training/intro.yml.html#265>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Container networking basics*

24.1.5.5 Container network drivers

See also:

- <https://avril2018.container.training/intro.yml.html#281>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Container network drivers*

24.1.5.6 Container network model

See also:

- <https://avril2018.container.training/intro.yml.html#288>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Container network model*

24.1.5.7 Service discovery with containers

See also:

- <https://avril2018.container.training/intro.yml.html#300>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Service discovery with containers*

24.1.5.8 Ambassadors

See also:

- <https://avril2018.container.training/intro.yml.html#322>
- <https://avril2018.container.training/intro.yml.html#10>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Ambassadors*

24.1.6 Chapter5 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#335>
- <https://avril2018.container.training/intro.yml.html#11>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.6.1 Local development workflow with Docker

See also:

- <https://avril2018.container.training/intro.yml.html#335>
- <https://avril2018.container.training/intro.yml.html#11>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Local development workflow with Docker*
 - *Objectives*
 - *Containerized local development environments*
 - *Working on the “namer” application*
 - *Looking at the code*

24.1.6.1.1 Objectives

At the end of this section, you will be able to:

- Share code between container and host.
- Use a simple local development workflow.

24.1.6.1.2 Containerized local development environments

See also:

- <https://avril2018.container.training/intro.yml.html#338>

We want to solve the following issues:

- “Works on my machine”
- “Not the same version”
- “Missing dependency”

By using Docker containers, we will get a consistent development environment.

24.1.6.1.3 Working on the “namer” application

See also:

- <https://avril2018.container.training/intro.yml.html#339>
- We have to work on some application whose code is at: <https://github.com/jpetazzo/namer>.
- What is it? We don’t know yet !
- Let’s download the code.

```
$ git clone https://github.com/jpetazzo/namer
```

24.1.6.1.4 Looking at the code

See also:

- <https://avril2018.container.training/intro.yml.html#340>

```
$ cd namer
$ ls -l
```

```
company_name_generator.rb
config.ru
docker-compose.yml
Dockerfile
Gemfile
```

24.1.6.2 Working with volumes

See also:

- <https://avril2018.container.training/intro.yml.html#373>
- <https://avril2018.container.training/intro.yml.html#11>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Working with volumes*
 - *Objectives*
 - *Working with volumes*
 - *Volumes are special directories in a container*
 - *Volumes bypass the copy-on-write system*
 - *Volumes can be shared across containers*
 - *Sharing app server logs with another container*
 - *Volumes exist independently of containers*
 - *Naming volumes*
 - *Using our named volumes*
 - *Using a volume in another container*
 - *Managing volumes explicitly*
 - *Migrating data with –volumes-from*
 - *Data migration in practice*
 - *Upgrading Redis*
 - *Testing the new Redis*
 - *Volumes lifecycle*
 - *Checking volumes defined by an image*
 - *Checking volumes used by a container*
 - *Sharing a single file*
 - *Volume plugins*
 - *Volumes vs. Mounts*
 - *–mount syntax*
 - *Section summary*

24.1.6.2.1 Objectives

See also:

- <https://avril2018.container.training/intro.yml.html#375>

At the end of this section, you will be able to:

- Create containers holding volumes.
- Share volumes across containers.
- Share a host directory with one or many containers.

24.1.6.2.2 Working with volumes

See also:

- <https://avril2018.container.training/intro.yml.html#376>

Docker volumes can be used to achieve many things, including:

- Bypassing the copy-on-write system to obtain **native disk I/O performance**.
- Bypassing copy-on-write to **leave some files out of docker commit**.
- Sharing a directory between multiple containers.
- Sharing a directory between the host and a container.
- Sharing a single file between the host and a container.

24.1.6.2.3 Volumes are special directories in a container

See also:

- <https://avril2018.container.training/intro.yml.html#377>

Volumes can be declared in two different ways.

- Within a Dockerfile, with a VOLUME instruction.

```
VOLUME /uploads
```

- On the command-line, with the -v flag for docker run.

```
$ docker run -d -v /uploads myapp
```

In both cases, /uploads (inside the container) will be a volume.

24.1.6.2.4 Volumes bypass the copy-on-write system

See also:

- <https://avril2018.container.training/intro.yml.html#378>

Volumes act as passthroughs to the host filesystem.

- The I/O performance on a volume is exactly the same as I/O performance on the Docker host.
- When you docker commit, the content of volumes is not brought into the resulting image.
- If a RUN instruction in a Dockerfile changes the content of a volume, those changes are not recorded neither.
- If a container is started with the –read-only flag, the volume will still be writable (unless the volume is a read-only volume).

24.1.6.2.5 Volumes can be shared across containers

See also:

- <https://avril2018.container.training/intro.yml.html#379>
- You can start a container with exactly the same volumes as another one.
- The new container will have the same volumes, in the same directories.
- They will contain exactly the same thing, and remain in sync.
- Under the hood, they are actually the same directories on the host anyway.
- This is done using the –volumes-from flag for docker run.
- We will see an example in the following slides.

24.1.6.2.6 Sharing app server logs with another container

See also:

- <https://avril2018.container.training/intro.yml.html#380>

Let's start a Tomcat container:

```
$ docker run --name webapp -d -p 8080:8080 -v /usr/local/tomcat/logs tomcat
```

Now, start an alpine container accessing the same volume:

```
$ docker run --volumes-from webapp alpine sh -c "tail -f /usr/local/tomcat/logs/*"
```

Then, from another window, send requests to our Tomcat container:

```
$ curl localhost:8080
```

24.1.6.2.7 Volumes exist independently of containers

See also:

- <https://avril2018.container.training/intro.yml.html#381>

If a container is stopped, its volumes still exist and are available.

Volumes can be listed and manipulated with docker volume subcommands:

```
$ docker volume ls
```

DRIVER	VOLUME NAME
local	5b0b65e4316da67c2d471086640e6005ca2264f3...
local	pgdata-prod
local	pgdata-dev
local	13b59c9936d78d109d094693446e174e5480d973...

Some of those volume names were explicit (pgdata-prod, pgdata-dev).

The others (the hex IDs) were generated automatically by Docker.

24.1.6.2.8 Naming volumes

See also:

- <https://avril2018.container.training/intro.yml.html#382>
- Volumes can be created without a container, then used in multiple containers.

Let's create a couple of volumes directly.

```
$ docker volume create webapps
```

```
webapps
```

```
$ docker volume create logs
```

```
logs
```

Volumes are not anchored to a specific path.

24.1.6.2.9 Using our named volumes

See also:

- <https://avril2018.container.training/intro.yml.html#383>
- Volumes are used with the -v option.
- When a host path does not contain a /, it is considered to be a volume name.

Let's start a web server using the two previous volumes.

```
docker run -d -p 1234:8080 -v logs:/usr/local/tomcat/logs -v webapps:/usr/local/
    ↪tomcat/webapps tomcat
```

```
Unable to find image 'tomcat:latest' locally
latest: Pulling from library/tomcat
ccl1a78bfd46b: Already exists
6861473222a6: Already exists
7e0b9c3b5ae0: Already exists
ae14ee39877a: Pull complete
8085c1b536f0: Pull complete
6e1431e84c0c: Pull complete
ca0e3df5a1fd: Pull complete
d2cb611ced6c: Pull complete
268dc3e43e66: Pull complete
79a7e8d254c7: Pull complete
5c848af92738: Pull complete
789b92e37607: Pull complete
Digest: sha256:a01c3ad30a211e742dabd74ff722374ab25c27b8d6162b210572a915305f1246
Status: Downloaded newer image for tomcat:latest
27cd9367df6a22034e3f79d55237fe928cd4af90a5e9261039d0236687ec121e
```

Check that it's running correctly:

```
$ curl localhost:1234
... (Tomcat tells us how happy it is to be up and running) ...
```

```
$ curl localhost:1234
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Apache Tomcat/8.5.31</title>
    <link href="favicon.ico" rel="icon" type="image/x-icon" />
    <link href="favicon.ico" rel="shortcut icon" type="image/x-icon" />
    <link href="tomcat.css" rel="stylesheet" type="text/css" />
  </head>

  <body>
    <div id="wrapper">
```

24.1.6.2.10 Using a volume in another container

See also:

- <https://avril2018.container.training/intro.yml.html#384>
- We will make changes to the volume from another container.
- In this example, we will run a text editor in the other container.
- (But this could be a FTP server, a WebDAV server, a Git receiver...)

Let's start another container using the webapps volume.

```
$ docker run -v webapps:/webapps -w /webapps -ti alpine vi ROOT/index.jsp
```

Vandalize the page, save, exit.

Then run curl localhost:1234 again to see your changes.

24.1.6.2.11 Managing volumes explicitly

See also:

- <https://avril2018.container.training/intro.yml.html#385>

In some cases, you want a specific directory on the host to be mapped inside the container:

- You want to manage storage and snapshots yourself.
- (With LVM, or a SAN, or ZFS, or anything else!)
- You have a separate disk with better performance (SSD) or resiliency (EBS) than the system disk, and you want to put important data on that disk.
- You want to share your source directory between your host (where the source gets edited) and the container (where it is compiled or executed).
- Wait, we already met the last use-case in our example development workflow! Nice.

```
$ docker run -d -v /path/on/the/host:/path/in/container image ...
```

24.1.6.2.12 Migrating data with –volumes-from

See also:

- <https://avril2018.container.training/intro.yml.html#386>

The –volumes-from option tells Docker to re-use all the volumes of an existing container.

- Scenario: migrating from Redis 2.8 to Redis 3.0.
- We have a container (myredis) running Redis 2.8.
- Stop the myredis container.
- Start a new container, using the Redis 3.0 image, and the –volumes-from option.
- The new container will inherit the data of the old one.
- Newer containers can use –volumes-from too.

24.1.6.2.13 Data migration in practice

See also:

- <https://avril2018.container.training/intro.yml.html#387>

Let's create a Redis container.

```
$ docker run -d --name redis28 redis:2.8
```

Connect to the Redis container and set some data.

```
$ docker run -ti --link redis28:redis alpine telnet redis 6379
```

Issue the following commands:

```
SET counter 42
INFO server
SAVE
QUIT
```

24.1.6.2.14 Upgrading Redis

See also:

- <https://avril2018.container.training/intro.yml.html#388>

Stop the Redis container.

```
$ docker stop redis28
```

Start the new Redis container.

```
$ docker run -d --name redis30 --volumes-from redis28 redis:3.0
```

24.1.6.2.15 Testing the new Redis

See also:

- <https://avril2018.container.training/intro.yml.html#389>

Connect to the Redis container and see our data.

```
docker run -ti --link redis30:redis alpine telnet redis 6379
```

Issue a few commands:

```
GET counter  
INFO server  
QUIT
```

24.1.6.2.16 Volumes lifecycle

See also:

- <https://avril2018.container.training/intro.yml.html#390>
- When you remove a container, its volumes are kept around.
- You can list them with docker volume ls.
- You can access them by creating a container with docker run -v.
- You can remove them with docker volume rm or docker system prune.

Ultimately, you are the one responsible for logging, monitoring, and backup of your volumes.

24.1.6.2.17 Checking volumes defined by an image

See also:

- <https://avril2018.container.training/intro.yml.html#391>

Wondering if an image has volumes? Just use docker inspect:

```
$ docker inspect training/datavol
```

```
[ {  
    "config": {  
        . . .  
        "Volumes": {  
            "/var/webapp": {}  
        },  
        . . .  
    } ]
```

24.1.6.2.18 Checking volumes used by a container

See also:

- <https://avril2018.container.training/intro.yml.html#392>

To look which paths are actually volumes, and to what they are bound, use docker inspect (again):

```
$ docker inspect <yourContainerID>
```

```
[{
  "ID": "<yourContainerID>",
  . . .
  "Volumes": {
    "/var/webapp": "/var/lib/docker/vfs/dir/
    ↪f4280c5b6207ed531efd4cc673ff620cef2a7980f747dbbccca001db61de04468"
  },
  "VolumesRW": {
    "/var/webapp": true
  },
}]
```

- We can see that our volume is present on the file system of the Docker host.

24.1.6.2.19 Sharing a single file

See also:

- <https://avril2018.container.training/intro.yml.html#393>

The same -v flag can be used to share a single file (instead of a directory). One of the most interesting examples is to share the Docker control socket.

```
$ docker run -it -v /var/run/docker.sock:/var/run/docker.sock docker sh
```

From that container, you can now run docker commands communicating with the Docker Engine running on the host. Try docker ps!

Since that container has access to the Docker socket, it has root-like access to the host.

24.1.6.2.20 Volume plugins

See also:

- <https://avril2018.container.training/intro.yml.html#394>
- <https://github.com/ClusterHQ/dvol>
- <https://github.com/rexray/rexray>
- <http://www.blockbridge.com/>
- <https://portworx.com/>

You can install plugins to manage volumes backed by particular storage systems, or providing extra features. For instance:

- dvol - allows to commit/branch/rollback volumes;
- Flocker, REX-Ray - create and manage volumes backed by an enterprise storage system (e.g. SAN or NAS), or by cloud block stores (e.g. EBS);
- Blockbridge, Portworx - provide distributed block store for containers;
- and much more!

24.1.6.2.21 Volumes vs. Mounts

See also:

- <https://avril2018.container.training/intro.yml.html#395>
- Since Docker 17.06, a new option is available: `--mount`.
- It offers a new, richer syntax to manipulate data in containers.
- It makes an explicit difference between:
 - volumes (identified with a unique name, managed by a storage plugin),
 - bind mounts (identified with a host path, not managed).
- The former `-v` / `--volume` option is still usable

24.1.6.2.22 --mount syntax

See also:

- <https://avril2018.container.training/intro.yml.html#396>

Binding a host path to a container path:

```
$ docker run --mount type=bind,source=/path/on/host,target=/path/in/container alpine
```

Mounting a volume to a container path:

```
$ docker run --mount source=myvolume,target=/path/in/container alpine
```

Mounting a tmpfs (in-memory, for temporary files)

```
$ docker run --mount type=tmpfs,destination=/path/in/container,tmpfs-size=1000000 ↵alpine
```

24.1.6.2.23 Section summary

See also:

- <https://avril2018.container.training/intro.yml.html#397>

We've learned how to:

- Create and manage volumes.
- Share volumes across containers.
- Share a host directory with one or many containers.

24.1.6.3 Compose for development stacks

See also:

- <https://avril2018.container.training/intro.yml.html#399>
- <https://avril2018.container.training/intro.yml.html#11>
- *Les conseils et formations de Jérôme Petazzoni*

Contents

- *Compose for development stacks*
 - *Compose for development stacks*
 - *What is Docker Compose ?*
 - *Compose overview*
 - *Checking if Compose is installed*
 - *Launching Our First Stack with Compose*
 - *Launching Our First Stack with Compose*
 - *Stopping the app*
 - *The docker-compose.yml file*
 - *Compose file versions*
 - *Containers in docker-compose.yml*
 - *Container parameters*
 - *Compose commands*
 - *Check container status*
 - *Cleaning up (1)*
 - *Cleaning up (2)*
 - *Special handling of volumes*
 - *Compose project name*
 - *Running two copies of the same app*

24.1.6.3.1 Compose for development stacks

See also:

- <https://avril2018.container.training/intro.yml.html#400>

Dockerfiles are great to build container images.

But what if we work with a complex stack made of multiple containers ?

Eventually, we will want to write some custom scripts and automation to build, run, and connect our containers together.

There is a better way: using **Docker Compose**.

In this section, you will use Compose to bootstrap a development environment

24.1.6.3.2 What is Docker Compose ?

See also:

- <https://avril2018.container.training/intro.yml.html#401>

Docker Compose (formerly known as fig) is an external tool.

Unlike the Docker Engine, it is written in Python. It's open source as well.

The general idea of Compose is to enable a very simple, powerful onboarding workflow:

- Checkout your code.
- Run docker-compose up.
- Your app is up and running !

24.1.6.3.3 Compose overview

See also:

- <https://avril2018.container.training/intro.yml.html#402>

This is how you work with Compose:

- You describe a set (or stack) of containers in a YAML file called docker-compose.yml.
- You run docker-compose up.
- Compose automatically pulls images, builds containers, and starts them.
- Compose can set up links, volumes, and other Docker options for you.
- Compose can run the containers in the background, or in the foreground.
- When containers are running in the foreground, their aggregated output is shown.

Before diving in, let's see a small example of Compose in action.

24.1.6.3.4 Checking if Compose is installed

See also:

- <https://avril2018.container.training/intro.yml.html#404>

If you are using the official training virtual machines, Compose has been pre-installed.

You can always check that it is installed by running:

```
$ docker-compose --version
```

24.1.6.3.5 Launching Our First Stack with Compose

See also:

- <https://avril2018.container.training/intro.yml.html#405>
- *docker-compose up*

First step: clone the source code for the app we will be working on.

```
$ cd
$ git clone git://github.com/jpetazzo/trainingwheels
...
$ cd trainingwheels
```

Second step: start your app.

```
$ docker-compose up
```

Watch Compose build and run your app with the correct parameters, including linking the relevant containers together.

```
## cat docker-compose.yml
```

```
version: "2"

services:
  www:
    build: www
    ports:
      - 8000:5000
    user: nobody
    environment:
      DEBUG: 1
    command: python counter.py
    volumes:
      - ./www:/src

  redis:
    image: redis
```

```
$ tree
```

```
.
├── docker-compose.yml
├── docker-compose.yml-ecs
└── ports.yml
    └── www
        ├── assets
        │   ├── css
        │   │   ├── bootstrap.min.css
        │   │   └── bootstrap-responsive.min.css
        │   └── js
        │       └── bootstrap.min.js
        ├── counter.py
        ├── Dockerfile
        └── templates
            ├── error.html
            └── index.html
```

5 directories, 10 files

```
$ docker-compose up
```

```
Creating network "trainingwheels_default" with the default driver
Building www
Step 1/8 : FROM python
latest: Pulling from library/python
ccla78bfd46b: Pull complete
6861473222a6: Pull complete
7e0b9c3b5ae0: Pull complete
3ec98735f56f: Pull complete
```

(continues on next page)

(continued from previous page)

```
9b311b87a021: Pull complete
048165938570: Pull complete
1ca3d78efb22: Pull complete
0f6c8999c3b7: Pull complete
5a85410f5000: Pull complete
Digest: sha256:52a2bd143faf6430b182b56a5fdeb70f26b8ca8fdbd40210c3ed8a8ee1eaba343
Status: Downloaded newer image for python:latest
--> 29d2f3226daf
Step 2/8 : RUN pip install flask
--> Running in 30e9159dd9dc
Collecting flask
    Downloading https://files.pythonhosted.org/packages/7f/e7/
→ 08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb4b/Flask-1.0.2-py2.py3-
→ none-any.whl (91kB)
Collecting itsdangerous>=0.24 (from flask)
    Downloading https://files.pythonhosted.org/packages/dc/b4/
→ a60bcd9a45c00f6d608d8975131ab3f25b22f2bcfe1dab221165194b2d4/itsdangerous-0.24.tar.
→ gz (46kB)
Collecting Jinja2>=2.10 (from flask)
    Downloading https://files.pythonhosted.org/packages/7f/ff/
→ ae64bacdfc95f27a016a7bed8e8686763ba4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-
→ none-any.whl (126kB)
Collecting click>=5.1 (from flask)
    Downloading https://files.pythonhosted.org/packages/34/c1/
→ 8806f99713ddb993c5366c362b2f908f18269f8d792aff1abfd700775a77(click-6.7-py2.py3-none-
→ any.whl (71kB)
Collecting Werkzeug>=0.14 (from flask)
    Downloading https://files.pythonhosted.org/packages/20/c4/
→ 12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243/Werkzeug-0.14.1-py2.
→ py3-none-any.whl (322kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10->flask)
    Downloading https://files.pythonhosted.org/packages/4d/de/
→ 32d741db316d8fdb7680822dd37001ef7a448255de9699ab4bfcbdf4172b/MarkupSafe-1.0.tar.gz
Building wheels for collected packages: itsdangerous, MarkupSafe
    Running setup.py bdist_wheel for itsdangerous: started
    Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
    Stored in directory: /root/.cache/pip/wheels/2c/4a/61/
→ 5599631c1554768c6290b08c02c72d7317910374ca602ff1e5
    Running setup.py bdist_wheel for MarkupSafe: started
    Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
    Stored in directory: /root/.cache/pip/wheels/33/56/20/
→ ebe49a5c612fffe1c5a632146b16596f9e64676768661e4e46
Successfully built itsdangerous MarkupSafe
Installing collected packages: itsdangerous, MarkupSafe, Jinja2, click, Werkzeug, flask
Successfully installed Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 click-6.7 flask-1.0.
→ 2 itsdangerous-0.24
Removing intermediate container 30e9159dd9dc
--> 715be459df83
Step 3/8 : RUN pip install gunicorn
--> Running in 27a29e572569
Collecting gunicorn
    Downloading https://files.pythonhosted.org/packages/55/cb/
→ 09fe80bddf30be86abfc06ccb1154f97d6c64bb87111de066a5fc9ccb937/gunicorn-19.8.1-py2.
→ py3-none-any.whl (112kB)
Installing collected packages: gunicorn
Successfully installed gunicorn-19.8.1
```

(continues on next page)

(continued from previous page)

```

Removing intermediate container 27a29e572569
--> cd78b2130321
Step 4/8 : RUN pip install redis
--> Running in 637a8b3cd24b
Collecting redis
  Downloading https://files.pythonhosted.org/packages/3b/f6/
  ↳ 7a76333cf0b9251ecf49efff635015171843d9b977e4ffcf59f9c4428052/redis-2.10.6-py2.py3-
  ↳ none-any.whl (64kB)
Installing collected packages: redis
Successfully installed redis-2.10.6
Removing intermediate container 637a8b3cd24b
--> 08766036473f
Step 5/8 : COPY . /src
--> 4de5b2a959d5
Step 6/8 : WORKDIR /src
Removing intermediate container 6013def61017
--> 54eb5e672592
Step 7/8 : CMD gunicorn --bind 0.0.0.0:5000 --workers 10 counter:app
--> Running in bab6ea1f334c
Removing intermediate container bab6ea1f334c
--> 585a2f6a0163
Step 8/8 : EXPOSE 5000
--> Running in 228ff16daa14
Removing intermediate container 228ff16daa14
--> d0ad402a2cc3
Successfully built d0ad402a2cc3
Successfully tagged trainingwheels_www:latest
WARNING: Image for service www was built because it did not already exist. To rebuild,
--> this image you must use `docker-compose build` or `docker-compose up --build`.
Pulling redis (redis:)...  

latest: Pulling from library/redis
4d0d76e05f3c: Pull complete
cfbf30a55ec9: Pull complete
82648e31640d: Pull complete
fb7ace35d550: Pull complete
497bf119bebf: Pull complete
89340f6074da: Pull complete
Digest: sha256:4aed8ea5a5fc4cf05c8d5341b4ae4a4f7c0f9301082a74f6f9a5f321140e0cd3
Status: Downloaded newer image for redis:latest
Creating trainingwheels_www_1 ... done
Creating trainingwheels_redis_1 ... done
Attaching to trainingwheels_redis_1, trainingwheels_www_1
redis_1 | 1:C 01 Jun 07:45:02.780 # o000o000o000o Redis is starting o000o000o000o
redis_1 | 1:C 01 Jun 07:45:02.780 # Redis version=4.0.9, bits=64, commit=00000000,
--> modified=0, pid=1, just started
redis_1 | 1:C 01 Jun 07:45:02.780 # Warning: no config file specified, using the
--> default config. In order to specify a config file use redis-server /path/to/redis.
--> conf
redis_1 | 1:M 01 Jun 07:45:02.782 * Running mode=standalone, port=6379.
redis_1 | 1:M 01 Jun 07:45:02.782 # WARNING: The TCP backlog setting of 511 cannot
--> be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
redis_1 | 1:M 01 Jun 07:45:02.782 # Server initialized
redis_1 | 1:M 01 Jun 07:45:02.782 # WARNING overcommit_memory is set to 0!
--> Background save may fail under low memory condition. To fix this issue add 'vm.
--> overcommit_memory = 1' to /etc/sysctl.conf and then
reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis_1 | 1:M 01 Jun 07:45:02.782 # WARNING you have Transparent Huge Pages (THP)
--> support enabled in your kernel. This will create latency and memory usage issues
--> with Redis. To fix this issue run the comma

```

(continues on next page)

(continued from previous page)

```
nd 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
redis_1 | 1:M 01 Jun 07:45:02.782 * Ready to accept connections
www_1 |   * Serving Flask app "counter" (lazy loading)
www_1 |   * Environment: production
www_1 |     WARNING: Do not use the development server in a production environment.
www_1 |     Use a production WSGI server instead.
www_1 |   * Debug mode: on
www_1 |   * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
www_1 |   * Restarting with stat
www_1 |   * Debugger is active!
www_1 |   * Debugger PIN: 313-495-332
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET / HTTP/1.1" 200 -
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET /assets/css/bootstrap.min.css HTTP/1.1" 200 -
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET /assets/css/bootstrap-responsive.min.css HTTP/1.1" 200 -
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET /assets/js/bootstrap.min.js HTTP/1.1" 200 -
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET /favicon.ico HTTP/1.1" 404 -
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET /favicon.ico HTTP/1.1" 404 -
www_1 | X.X.X.X -- [01/Jun/2018 07:49:36] "GET /favicon.ico HTTP/1.1" 404 -
```

24.1.6.3.6 Launching Our First Stack with Compose

See also:

- <https://avril2018.container.training/intro.yml.html#406>

Verify that the app is running at <http://<yourHostIP>:8000>.



24.1.6.3.7 Stopping the app

See also:

- <https://avril2018.container.training/intro.yml.html#407>

When you hit ^C, Compose tries to gracefully terminate all of the containers.

After ten seconds (or if you press ^C again) it will forcibly kill them.

```
^CGracefully stopping... (press Ctrl+C again to force)
Stopping trainingwheels_www_1 ... done
Stopping trainingwheels_redis_1 ... done
```

24.1.6.3.8 The docker-compose.yml file

See also:

- <https://avril2018.container.training/intro.yml.html#408>

Here is the file used in the demo:

```
version: "2"
services:
  www:
    build: www
    ports:
      - 8000:5000
    user: nobody
    environment:
      DEBUG: 1
    command: python counter.py
    volumes:
      - ./www:/src
  redis:
    image: redis
```

```
$ cat www/Dockerfile
```

```
FROM python
RUN pip install flask
RUN pip install gunicorn
RUN pip install redis
COPY . /src
WORKDIR /src
CMD gunicorn --bind 0.0.0.0:5000 --workers 10 counter:app
EXPOSE 5000
```

24.1.6.3.9 Compose file versions

See also:

- <https://avril2018.container.training/intro.yml.html#409>
- <https://docs.docker.com/compose/compose-file/>

Version 1 directly has the various containers (www, redis...) at the top level of the file.

Version 2 has multiple sections:

- version is mandatory and should be “2”.
- services is mandatory and corresponds to the content of the version 1 format.
- networks is optional and indicates to which networks containers should be connected. (By default, containers will be connected on a private, per-app network.)
- volumes is optional and can define volumes to be used and/or shared by the containers.

Version 3 adds support for deployment options (scaling, rolling updates, etc.)

24.1.6.3.10 Containers in docker-compose.yml

See also:

- <https://avril2018.container.training/intro.yml.html#410>

Each service in the YAML file must contain either build, or image.

- *build* indicates a path containing a Dockerfile.
- *image* indicates an image name (local, or on a registry).
- If both are specified, an image will be built from the build directory and named image.

The other parameters are optional.

They encode the parameters that you would typically add to docker run.

Sometimes they have several minor improvements.

24.1.6.3.11 Container parameters

See also:

- <https://avril2018.container.training/intro.yml.html#411>
- <https://docs.docker.com/compose/compose-file/>
- *compose-file*
- command indicates what to run (like CMD in a Dockerfile).
- ports translates to one (or multiple) -p options to map ports. You can specify local ports (i.e. x:y to expose public port x).
- volumes translates to one (or multiple) -v options. You can use relative paths here.

For the full list, check: <https://docs.docker.com/compose/compose-file/>

24.1.6.3.12 Compose commands

See also:

- <https://avril2018.container.training/intro.yml.html#412>

We already saw docker-compose up, but another one is **docker-compose build**

It will execute docker build for all containers mentioning a build path.

It can also be invoked automatically when starting the application:

```
docker-compose up --build
```

Another common option is to start containers in the background:

```
docker-compose up -d
```

24.1.6.3.13 Check container status

See also:

- <https://avril2018.container.training/intro.yml.html#413>

It can be tedious to check the status of your containers with docker ps, especially when running multiple apps at the same time.

Compose makes it easier; with docker-compose ps you will see only the status of the containers of the current stack:

\$ docker-compose ps				
	Name	Command	State	
Ports				↳

redis	trainingwheels_redis_1	docker-entrypoint.sh redis ...	Up	6379/tcp
www	trainingwheels_www_1	python counter.py	Up	0.0.0.0:8000->5000/
tcp				

24.1.6.3.14 Cleaning up (1)

See also:

- <https://avril2018.container.training/intro.yml.html#414>

If you have started your application in the background with Compose and want to stop it easily, you can use the kill command:

\$ docker-compose kill	

Likewise, docker-compose rm will let you remove containers (after confirmation):

\$ docker-compose rm	

```
Going to remove trainingwheels_redis_1, trainingwheels_www_1
Are you sure? [yN] y
Removing trainingwheels_redis_1...
Removing trainingwheels_www_1...
```

24.1.6.3.15 Cleaning up (2)

See also:

- <https://avril2018.container.training/intro.yml.html#415>

Alternatively, docker-compose down will stop and remove containers.

It will also remove other resources, like networks that were created for the application.

\$ docker-compose down	

```
Stopping trainingwheels_www_1 ... done
Stopping trainingwheels_redis_1 ... done
Removing trainingwheels_www_1 ... done
Removing trainingwheels_redis_1 ... done
```

24.1.6.3.16 Special handling of volumes

See also:

- <https://avril2018.container.training/intro.yml.html#416>

Compose is smart. If your container uses volumes, when you restart your application, Compose will create a new container, but carefully re-use the volumes it was using previously.

This makes it easy to upgrade a stateful service, by pulling its new image and just restarting your stack with Compose.

24.1.6.3.17 Compose project name

See also:

- <https://avril2018.container.training/intro.yml.html#417>
- When you run a Compose command, Compose infers the “project name” of your app.
- By default, the “project name” is the name of the current directory.
- For instance, if you are in /home/zelda/src/ocarina, the project name is ocarina.
- All resources created by Compose are tagged with this project name.
- The project name also appears as a prefix of the names of the resources.
- E.g. in the previous example, service www will create a container ocarina_www_1.
- The project name can be overridden with docker-compose -p.

24.1.6.3.18 Running two copies of the same app

See also:

- <https://avril2018.container.training/intro.yml.html#418>

If you want to run two copies of the same app simultaneously, all you have to do is to make sure that each copy has a different project name.

You can:

- copy your code in a directory with a different name
- start each copy with **docker-compose -p myprojname up**

Each copy will run in a different network, totally isolated from the other.

This is ideal to debug regressions, do side-by-side comparisons, etc.

24.1.6.4 Managing hosts with Docker Machine

See also:

- <https://avril2018.container.training/intro.yml.html#420>
- <https://avril2018.container.training/intro.yml.html#11>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.7 Chapter6 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#427>
- <https://avril2018.container.training/intro.yml.html#12>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.8 Chapter7 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#493>
- <https://avril2018.container.training/intro.yml.html#13>
- *Les conseils et formations de Jérôme Petazzoni*

24.1.9 Chapter8 Avril 2018

See also:

- <https://avril2018.container.training/intro.yml.html#583>
- <https://avril2018.container.training/intro.yml.html#14>
- *Les conseils et formations de Jérôme Petazzoni*

24.2 Les conseils et formations de Jérôme Petazzoni

See also:

- <https://avril2018.container.training/>
- *Avril 2018 container training from Jérôme Petazzoni*
- <https://github.com/jpetazzo/container.training>
- <https://jpetazzo.github.io/2018/03/28/containers-par-ou-commencer/>
- <https://github.com/jpetazzo>
- <https://twitter.com/jpetazzo>
- <https://twitter.com/jeremygarrouste>
- <https://github.com/jpetazzo/container.training>

- <https://training.play-with-docker.com>
- <http://paris.container.training/intro.html>
- <http://paris.container.training/kube.html>
- https://www.youtube.com/playlist?list=PLBAFXs0YjviLgqTum8MkspG_8VzGl6C07 (Docker)
- https://www.youtube.com/playlist?list=PLBAFXs0YjviLrsyydCzxWrIP_1-wkcSHS (Kubernetes)

Contents

- *Les conseils et formations de Jérôme Petazzoni*
 - *Se former, seul ou accompagné*
 - *Jérôme Petazzoni Container training*
 - *Jérémy Garrouste*
 - *Les slides de la formation d'avril 2018*

24.2.1 Se former, seul ou accompagné

La communauté Docker est extrêmement riche en tutoriels divers pour démarrer et aller plus loin.

Je recommande particulièrement les **labs** disponibles sur training.play-with-docker.com

Si vous préférez être formé en personne, c'est aussi possible !

Publicité bien ordonnée commence par soi-même : en avril, j'organise deux formations à Paris avec Jérémy Garrouste.

- Le 11 et 12 avril, *Introduction aux containers : de la pratique aux bonnes pratiques*.
- Le 13 avril, *Introduction à l'orchestration : Kubernetes par l'exemple*

La première formation vous permettra d'être à même d'accomplir les deux premières étapes décrites dans le plan exposé plus haut.

La seconde formation vous permettra d'aborder les étapes 3 et 4.

Si vous voulez vous faire une idée de la qualité du contenu de ces formations, vous pouvez consulter des vidéos et slides de formations précédentes, par exemple :

- [journée d'introduction à Docker](#)
- [demi-journée d'introduction à Kubernetes](#)

Ces vidéos sont en anglais, mais les formations que je vous propose à Paris en avril sont en français (le support de formation, lui, reste en anglais).

Vous pouvez trouver d'autres vidéos, ainsi qu'une collection de supports (slides etc.) sur <http://container.training/>.

Cela vous permettra de juger au mieux si ces formations sont adaptées à votre besoin !

24.2.2 Jérôme Petazzoni Container training

See also:

- <https://github.com/jpetazzo/container.training>
- <http://container.training/>

24.2.3 Jérémie Garrouste

See also:

- <https://twitter.com/jeremygarrouste>

24.2.4 Les slides de la formation d'avril 2018

See also:

<https://avril2018.container.training/>

24.3 Tutoriels Docker pour Windows

See also:

- <https://docs.docker.com/docker-for-windows/>

Contents

- *Tutoriels Docker pour Windows*
 - *Installation*
 - *docker –version*
 - *docker-compose –version*
 - *docker-machine –version*
 - *notary version*
 - *Binaires docker sous Windows 10*
 - *Where to go next*

24.3.1 Installation

See also:

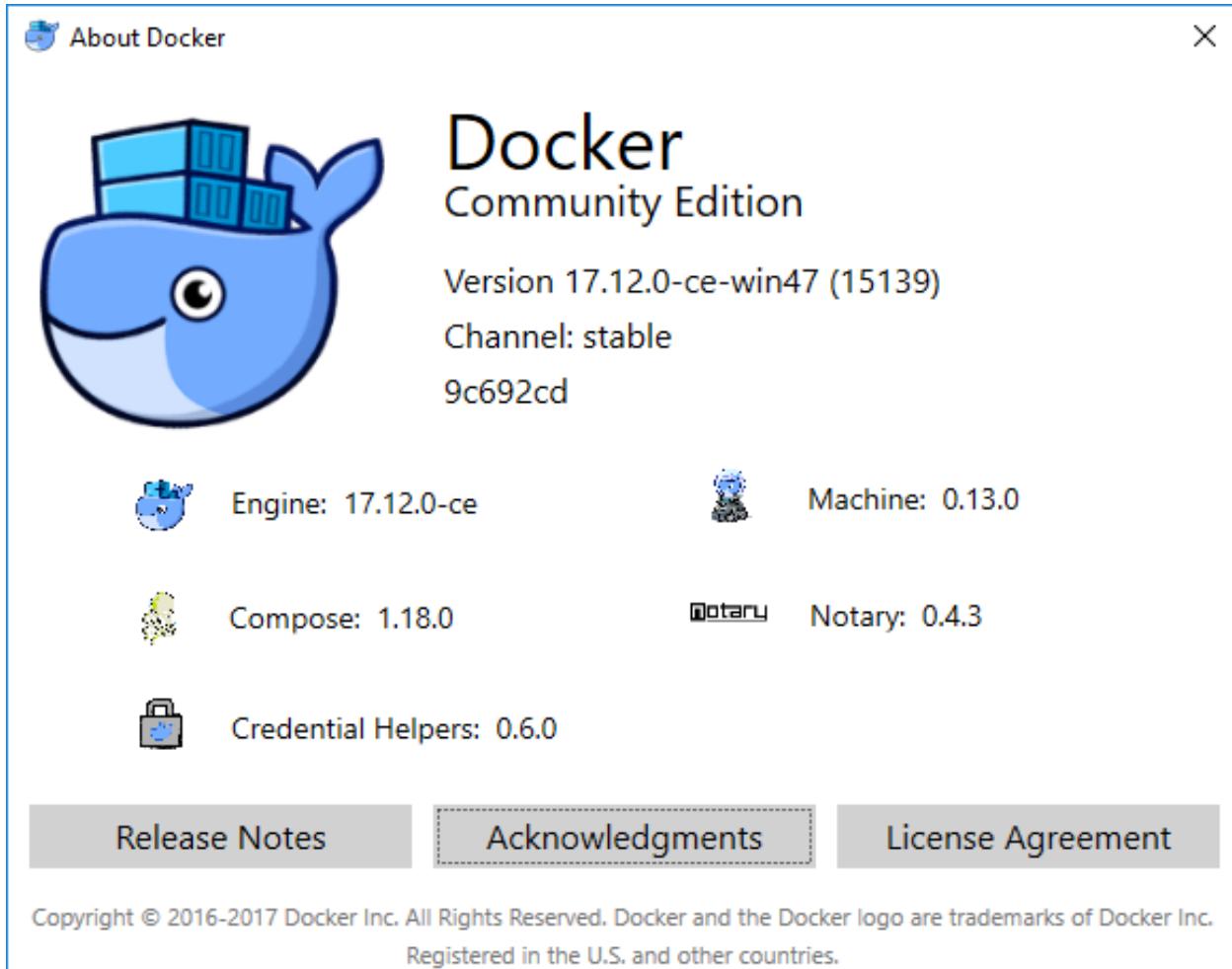
- <https://docs.docker.com/docker-for-windows/install/>
- <https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe>
- <https://nickjanetakis.com/blog/setting-up-docker-for-windows-and-wsl-to-work-flawlessly>

Installation de “Docker for windows” quand on a une machine sous Windows 10.

24.3.2 docker –version

```
docker --version
```

```
Docker version 17.12.0-ce, build c97c6d6
```



24.3.3 docker-compose –version

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker>tutorial_docker>docker-compose --version
```

```
docker-compose version 1.18.0, build 8dd22a96
```

24.3.4 docker-machine –version

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker>tutorial_docker>docker-machine --version
```

```
docker-machine version 0.13.0, build 9ba6da9
```

24.3.5 notary version

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker>notary version
```

```
notary
Version: 0.4.3
Git commit: 9211198
```

24.3.6 Binaires docker sous Windows 10

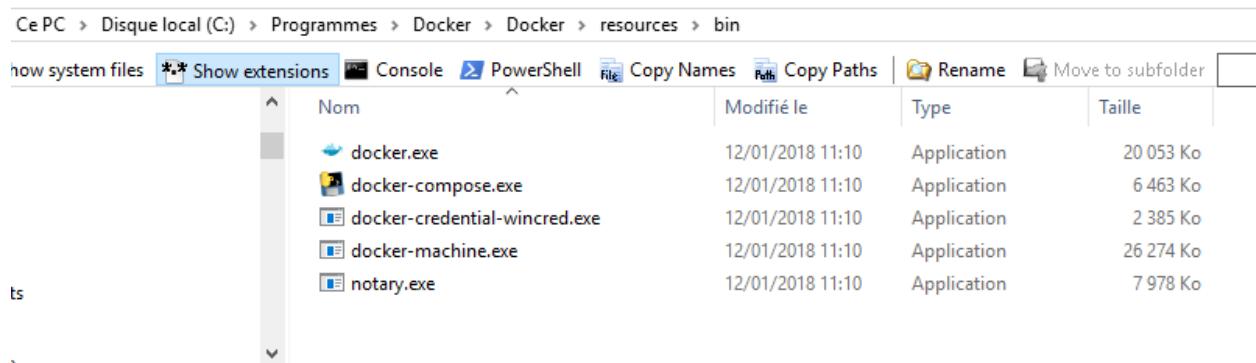


Fig. 1: Binaires docker sous Windows 10

24.3.7 Where to go next

See also:

- <https://docs.docker.com/get-started/>
- <https://github.com/docker/labs/>
- <https://docs.docker.com/engine/reference/commandline/docker/>
- <https://blog.docker.com/2017/01/whats-new-in-docker-1-13/>

- Try out the walkthrough at [Get Started](#).
- Dig in deeper with [Docker Labs](#) example walkthroughs and source code.
- For a summary of Docker command line interface (CLI) commands, see the [Docker CLI Reference Guide](#).
- Check out the blog post [Introducing Docker 1.13.0](#).

24.4 Get started (<https://docs.docker.com/get-started/>)

See also:

- <https://docs.docker.com/get-started/>
- *Tutoriels Docker pour Windows*

Contents

- *Get started (<https://docs.docker.com/get-started/>)*
 - *docker run hello-world*
 - *docker -version*
 - *Conclusion*
 - *Parts*

24.4.1 docker run hello-world

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker>docker run hello-
←world
```

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
\$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
<https://cloud.docker.com/>

For more examples and ideas, visit:
<https://docs.docker.com/engine/userguide/>

24.4.2 docker --version

```
Y:\projects_id3\P5N001\xLOGCA135_tutorial_docker\tutorial_docker>docker --version
```

```
Docker version 17.12.0-ce, build c97c6d6
```

24.4.3 Conclusion

The unit of scale being an individual, portable executable has vast implications.

It means CI/CD can push updates to any part of a distributed application, system dependencies are not an issue, and resource density is increased.

Orchestration of scaling behavior is a matter of spinning up new executables, not new VM hosts.

We'll be learning about all of these things, but first let's learn to walk.

24.4.4 Parts

24.4.4.1 Get started Part2 : Containers

See also:

- <https://docs.docker.com/get-started/part2/>
- <https://hub.docker.com/>
- <https://hub.docker.com/u/id3pvergain/>
- *Tutoriels Docker pour Windows*

Contents

- *Get started Part2 : Containers*
 - *Prérequis*
 - *Build the app: docker build -t friendlyhello .*
 - *docker images*
 - *Run the app: docker run -p 4000:80 friendlyhello*
 - *docker container ls*
 - *docker container stop 06193b763075*
 - *Tag the image: docker tag friendlyhello id3pvergain/get-started:part2*
 - *Publish the image*
 - *Pull and run the image from the remote repository*

24.4.4.1.1 Prérequis

Ne pas oublier de démarrer le serveur docker.

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\get_
→started\part2>docker build -t friendlyhello .
```

```
error during connect: Post http://%2F%2F.%2Fpipe%2Fdocker_engine/v1.35/build?
→buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&
→cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&
→networkmode=default&rm=1&
→session=503be270159342059d8cbfa34d94c9f1e312558a1dcef2ef4369cb0b440ad6a3&shmsize=0&
→t=friendlyhello&targt=&ulimits=null:
open //./pipe/docker_engine: Le fichier spécifié est introuvable.
In the default daemon configuration on Windows, the docker client
must be run elevated to connect.
This error may also indicate that the docker daemon is not running.
```

24.4.4.1.2 Build the app: docker build -t friendlyhello .

```
:: docker build -t friendlyhello .
```

```
Sending build context to Docker daemon 7.168kB
Step 1/7 : FROM python:2.7-slim
2.7-slim: Pulling from library/python
c4bb02b17bb4: Pull complete
c5c896dce5ee: Pull complete
cf210b898cc6: Pull complete
5117cef49bdb: Pull complete
Digest: sha256:22112f2295fe9ea84b72e5344af73a2580a47b1014a1f4c58eccf6095b7ea18f
Status: Downloaded newer image for python:2.7-slim
--> 4fd30fc83117
Step 2/7 : WORKDIR /app
Removing intermediate container 8ed2ad0d0958
--> 7400c8709865
Step 3/7 : ADD . /app
--> 728e5124216a
Step 4/7 : RUN pip install --trusted-host pypi.python.org -r requirements.txt
--> Running in 847d00a0831e
Collecting Flask (from -r requirements.txt (line 1))
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting Redis (from -r requirements.txt (line 2))
  Downloading redis-2.10.6-py2.py3-none-any.whl (64kB)
Collecting itsdangerous>=0.21 (from Flask->-r requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2>=2.4 (from Flask->-r requirements.txt (line 1))
  Downloading Jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting Werkzeug>=0.7 (from Flask->-r requirements.txt (line 1))
  Downloading Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
Collecting click>=2.0 (from Flask->-r requirements.txt (line 1))
  Downloading click-6.7-py2.py3-none-any.whl (71kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->Flask->-r requirements.txt (line 1))
  Downloading MarkupSafe-1.0.tar.gz
Building wheels for collected packages: itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for itsdangerous: started
  Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/fc/a8/66/
→24d655233c757e178d45dea2de22a04c6d92766abfb741129a
  Running setup.py bdist_wheel for MarkupSafe: started
```

(continues on next page)

(continued from previous page)

```

Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/88/a7/30/
→e39a54a87bcbe25308fa3ca64e8ddc75d9b3e5afa21ee32d57
Successfully built itsdangerous MarkupSafe
Installing collected packages: itsdangerous, MarkupSafe, Jinja2, Werkzeug, click, Flask, Redis
Successfully installed Flask-0.12.2 Jinja2-2.10 MarkupSafe-1.0 Redis-2.10.6 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 847d00a0831e
--> 3dc371ea405c
Step 5/7 : EXPOSE 80
--> Running in 0f4b33dbfcfd0
Removing intermediate container 0f4b33dbfcfd0
--> d1d59914b22b
Step 6/7 : ENV NAME World
--> Running in a742b8e9bddb
Removing intermediate container a742b8e9bddb
--> b79587f955c5
Step 7/7 : CMD ["python", "app.py"]
--> Running in f9c7ee2841c0
Removing intermediate container f9c7ee2841c0
--> ed5b70620e49
Successfully built ed5b70620e49
Successfully tagged friendlyhello:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions.
It is recommended to double check and reset permissions for sensitive files and directories.

```

24.4.4.1.3 docker images

docker images

REPOSITORY	TAG	IMAGE ID	CREATED
friendlyhello	latest	ed5b70620e49	10 minutes ago
wordpress	latest	28084cde273b	6 days ago
centos	latest	ff426288ea90	6 days ago
nginx	latest	3f8a4339aadd	2 weeks ago
python	2.7-slim	4fd30fc83117	4 weeks ago
hello-world	latest	f2a91732366c	7 weeks ago
docker4w/nsenter-dockerd	latest	cae870735e91	2 months ago

24.4.4.1.4 Run the app: docker run -p 4000:80 friendlyhello

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker>docker run -p 4000:80 friendlyhello
```

```
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```



Hello World!

Hostname: 16eca9f1274e

Visits: *cannot connect to Redis, counter disabled*

Fig. 2: <http://localhost:4000/>

24.4.4.1.5 docker container ls

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker>docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
	PORTS			NAMES
06193b763075	friendlyhello	"python app.py"	41 minutes ago	Up 41m
↳ minutes	0.0.0.0:4000->80/tcp	boring_goodall		

24.4.4.1.6 docker container stop 06193b763075

```
docker container stop 06193b763075
```

```
06193b763075
```

24.4.4.1.7 Tag the image: docker tag friendlyhello id3pvergain/get-started:part2

```
docker tag friendlyhello id3pvergain/get-started:part2
```

24.4.4.1.8 Publish the image

```
docker push id3pvergain/get-started:part2
```

```
The push refers to repository [docker.io/id3pvergain/get-started]
af88fcfe37d7: Pushed
b13ed1abc5b3: Pushed
150ac820623b: Pushed
94b0b6f67798: Mounted from library/python
e0c374004259: Mounted from library/python
56ee7573ea0f: Mounted from library/python
cfce7a8ae632: Mounted from library/python
part2: digest:sha256:1afb795959667db38cc58581d8d455ce10eff78be3cce18560ba887fb6f8c920 size: 1788
```

Once complete, the results of this upload are publicly available. If you log in to Docker Hub, you will see the new image there, with its pull command.

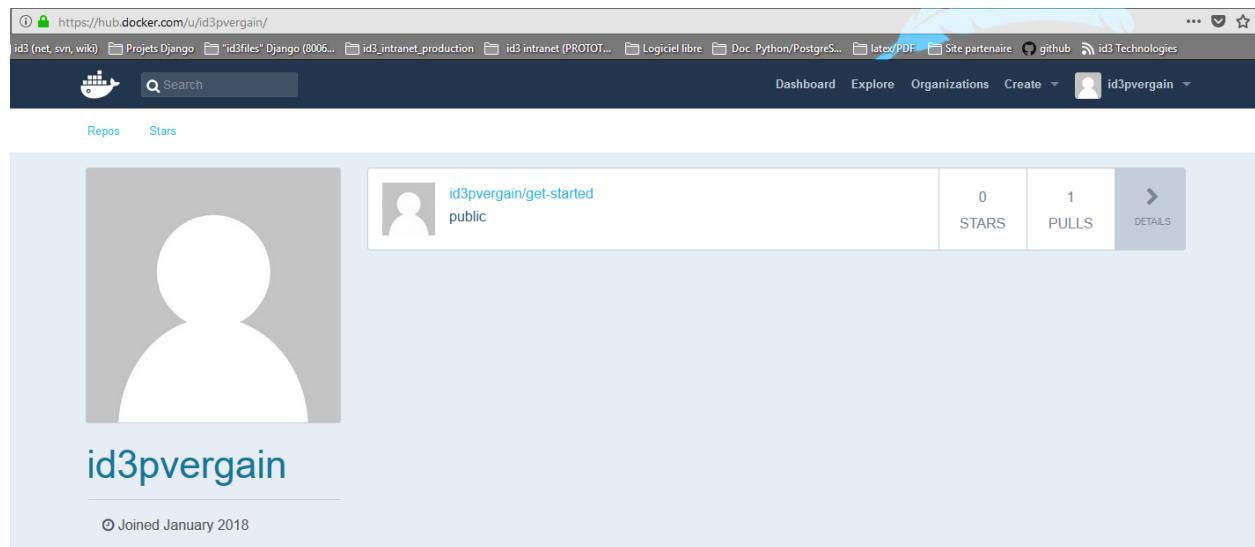


Fig. 3: <https://hub.docker.com/u/id3pvergain/>

24.4.4.1.9 Pull and run the image from the remote repository

See also:

- <https://docs.docker.com/get-started/part2/#pull-and-run-the-image-from-the-remote-repository>

From now on, you can use docker run and run your app on any machine with this command:

```
docker run -p 4000:80 id3pvergain/get-started:part2
```

If the image isn't available locally on the machine, Docker will pull it from the repository.

Here is a list of the basic Docker commands from this page, and some related ones if you'd like to explore a bit before moving on.

```
docker build -t friendlyhello . # Create image using this directory's Dockerfile
docker run -p 4000:80 friendlyhello # Run "friendlyname" mapping port 4000 to 80
docker run -d -p 4000:80 friendlyhello # Same thing, but in detached mode
```

(continues on next page)

(continued from previous page)

```

docker container ls                                # List all running containers
docker container ls -a                            # List all containers, even those not running
docker container stop <hash>                      # Gracefully stop the specified container
docker container kill <hash>                      # Force shutdown of the specified container
docker container rm <hash>                        # Remove specified container from this machine
docker container rm $(docker container ls -a -q)    # Remove all containers
docker image ls -a                                # List all images on this machine
docker image rm <image id>                      # Remove specified image from this machine
docker image rm $(docker image ls -a -q)          # Remove all images from this machine
docker login                                         # Log in this CLI session using your Docker credentials
docker tag <image> username/repository:tag        # Tag <image> for upload to registry
docker push username/repository:tag                 # Upload tagged image to registry
docker run username/repository:tag                  # Run image from a registry

```

24.4.4.2 Get started Part3 : services

See also:

- <https://docs.docker.com/get-started/part3/>

Contents

- *Get started Part3 : services*
 - *Prerequisites*
 - *Introduction*
 - *About services*
 - *Your first docker-compose.yml file*
 - *Run your new load-balanced app*
 - *docker swarm init*
 - *docker stack deploy -c docker-compose.yml getstartedlab*
 - *docker service ls*
 - *docker service ps getstartedlab_web*
 - *docker container ls -q*
 - *Sous WSL (Windows Subsystem Linux)*
 - *Scale the app*
 - *Take down the app (docker stack rm getstartedlab)*
 - *Take down the swarm (docker swarm leave --force)*

24.4.4.2.1 Prerequisites

Be sure your image works as a deployed container. Run this command, slotting in your info for username, repo, and tag:

```
docker run -p 80:80 id3pvergain/get-started:part2
```

then visit <http://localhost/>.

24.4.4.2.2 Introduction

In part 3, we scale our application and enable load-balancing.

To do this, we must go one level up in the hierarchy of a distributed application: the service.

- Stack
- Services (you are here)
- Container (covered in part 2)

24.4.4.2.3 About services

In a distributed application, different pieces of the app are called “services.” For example, if you imagine a video sharing site, it probably includes a service for storing application data in a database, a service for video transcoding in the background after a user uploads something, a service for the front-end, and so on.

Services are really just “containers in production.” A service only runs one image, but it codifies the way that image runs—what ports it should use, how many replicas of the container should run so the service has the capacity it needs, and so on.

Scaling a service changes the number of container instances running that piece of software, assigning more computing resources to the service in the process.

Luckily it’s very easy to define, run, and scale services with the Docker platform – just write a docker-compose.yml file.

24.4.4.2.4 Your first docker-compose.yml file

A docker-compose.yml file is a YAML file that defines how Docker containers should behave in production.

Save this file as docker-compose.yml wherever you want. Be sure you have pushed the image you created in Part 2 to a registry, and update this .yml by replacing username/repo:tag with your image details.

```

1 version: "3"
2 services:
3   web:
4     # replace username/repo:tag with your name and image details
5     image: id3pvergain/get-started:part2
6     deploy:
7       replicas: 3
8       resources:
9         limits:
10           cpus: "0.1"
11           memory: 50M
12       restart_policy:
13         condition: on-failure
14     ports:
15       - "80:80"
16     networks:

```

(continues on next page)

(continued from previous page)

```
17      - webnet
18 networks:
19   webnet:
```

This docker-compose.yml file tells Docker to do the following:

- Pull the image we uploaded in step 2 from the registry.
- Run 5 instances of that image as a service called web, limiting each one to use, at most, 10% of the CPU (across all cores), and 50MB of RAM.
- Immediately restart containers if one fails.
- Map port 80 on the host to web's port 80.
- Instruct web's containers to share port 80 via a load-balanced network called webnet. (Internally, the containers themselves will publish to web's port 80 at an ephemeral port.)
- Define the webnet network with the default settings (which is a load-balanced overlay network).

24.4.4.2.5 Run your new load-balanced app

24.4.4.2.6 docker swarm init

Before we can use the docker stack deploy command we'll first run:

```
docker swarm init
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\get_
→started\part3> docker swarm init
```

```
Swarm initialized: current node (pnbt8079jvn6eceltf17kysp) is now a manager.
```

To add a worker to this swarm, run the following command:

```
    docker swarm join --token SWMTKN-1-
→24yfg27ko4ma40mgips1yn5syhcs6fmcc7jesi7rwq56a9volj-4152plyrb8p316fpnbmqaaa7x 192.
→168.65.3:2377
```

```
To add a manager to this swarm, run 'docker swarm join-token manager' and follow the_
→instructions.
```

24.4.4.2.7 docker stack deploy -c docker-compose.yml getstartedlab

Now let's run it. You have to give your app a name. Here, it is set to *getstartedlab*:

```
docker stack deploy -c docker-compose.yml getstartedlab
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\get_
→started\part3> docker stack deploy -c docker-compose.yml getstartedlab
```

```
Creating network getstartedlab_webnet
Creating service getstartedlab_web
```

24.4.4.2.8 docker service ls

Our single service stack is running 5 container instances of our deployed image on one host. Let's investigate.

Get the service ID for the one service in our application:

```
docker service ls
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\get_
→started\part3> docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE
PORTS				
tzjfv6o4bpzb	getstartedlab_web	replicated	5/5	
→id3pvergain/get-started:part2	*:80→80/tcp			

You'll see output for the web service, prepended with your app name. If you named it the same as shown in this example, the name will be getstartedlab_web. The service ID is listed as well, along with the number of replicas, image name, and exposed ports.

A single container running in a service is called a task.

Tasks are given unique IDs that numerically increment, up to the number of replicas you defined in docker-compose.yml.

List the tasks for your service:

24.4.4.2.9 docker service ps getstartedlab_web

```
docker service ps getstartedlab_web
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\get_
→started\part3> docker service ps getstartedlab_web
```

ID	NAME	IMAGE	NODE	PORTS
DESIRED STATE	CURRENT STATE	ERROR		
qx6cvv7knp0m	getstartedlab_web.1	id3pvergain/get-started:part2	linuxkit-	
→00155d280a10	Running	Running 31 minutes ago		
z9m5tsjo75pz	getstartedlab_web.2	id3pvergain/get-started:part2	linuxkit-	
→00155d280a10	Running	Running 31 minutes ago		
kv05oigiytf	getstartedlab_web.3	id3pvergain/get-started:part2	linuxkit-	
→00155d280a10	Running	Running 31 minutes ago		
as0f73cwv518	getstartedlab_web.4	id3pvergain/get-started:part2	linuxkit-	
→00155d280a10	Running	Running 31 minutes ago		
w4qqxjhsqwx3	getstartedlab_web.5	id3pvergain/get-started:part2	linuxkit-	
→00155d280a10	Running	Running 31 minutes ago		

24.4.4.2.10 docker container ls -q

Tasks also show up if you just list all the containers on your system, though that will not be filtered by service:

```
docker container ls -q
```

```
c31e71b41bdb
8780b68999cf
4ead2b07d319
473d75fd76f2
cae7ae5c659b
f45453da50cf
b47fd081642e
```

24.4.4.2.11 Sous WSL (Windows Subsystem Linux)

```
pvergain@uc026:/mnt/c/Users/pvergain/Documents$ which curl
```

```
/usr/bin/curl
```

```
pvergain@uc026:/etc/apt$ curl http://localhost
```

```
<h3>Hello World!</h3><b>Hostname:</b> f45453da50cf<br/><b>Visits:</b> <i>cannot ↳ connect to Redis, counter disabled</i>
```

24.4.4.2.12 Scale the app

You can scale the app by changing the replicas value in docker-compose.yml, saving the change, and re-running the docker stack deploy command:

```
docker stack deploy -c docker-compose.yml getstartedlab
```

Docker will do an in-place update, no need to tear the stack down first or kill any containers.

Now, re-run docker container ls -q to see the deployed instances reconfigured. If you scaled up the replicas, more tasks, and hence, more containers, are started.

24.4.4.2.13 Take down the app (docker stack rm getstartedlab)

Take the app down with docker stack rm:

```
docker stack rm getstartedlab
```

```
Removing service getstartedlab_web
Removing network getstartedlab_webnet
```

24.4.4.2.14 Take down the swarm (docker swarm leave --force)

```
docker swarm leave --force
```

```
Node left the swarm.
```

It's as easy as that to stand up and scale your app with Docker. You've taken a huge step towards learning how to run containers in production. Up next, you will learn how to run this app as a bonafide swarm on a cluster of Docker machines.

To recap, while typing `docker run` is simple enough, the true implementation of a container in production is running it as a service.

Services codify a container's behavior in a Compose file, and this file can be used to scale, limit, and redeploy our app.

Changes to the service can be applied in place, as it runs, using the same command that launched the service: **docker stack deploy**.

Some commands to explore at this stage:

```
docker stack ls                                     # List stacks or apps
docker stack deploy -c <composefile> <appname>    # Run the specified Compose file
docker service ls                                  # List running services associated with an app
docker service ps <service>                        # List tasks associated with an app
docker inspect <task or container>                # Inspect task or container
docker container ls -q                             # List container IDs
docker stack rm <appname>                         # Tear down an application
docker swarm leave --force                         # Take down a single node swarm from the manager
```

24.4.4.3 Get started Part4 : swarms

See also:

- <https://docs.docker.com/get-started/part4/>

Contents

- *Get started Part4 : swarms*
 - *Introduction*
 - *Understanding Swarm clusters*
 - *Set up your swarm*
 - *Encore Bloqué*
 - * *Solution*

24.4.4.3.1 Introduction

In *part 3*, you took an app you wrote in *part 2*, and defined how it should run in production by turning it into a service, scaling it up 5x in the process.

Here in part 4, you deploy this application onto a cluster, running it on multiple machines.

Multi-container, multi-machine applications are made possible by joining multiple machines into a *Dockerized cluster* called a **swarm**.

24.4.4.3.2 Understanding Swarm clusters

A swarm is a group of machines that are running Docker and joined into a cluster. After that has happened, you continue to run the Docker commands you're used to, but now they are executed on a cluster by a **swarm manager**.

The machines in a swarm can be physical or virtual. After joining a swarm, they are referred to as **nodes**.

Swarm managers can use several strategies to run containers, such as *emptiest node* – which fills the least utilized machines with containers. Or *global*, which ensures that each machine gets exactly one instance of the specified container. You instruct the swarm manager to use these strategies in the Compose file, just like the one you have already been using.

Swarm managers are the only machines in a swarm that can execute your commands, or authorize other machines to join the swarm as workers. Workers are just there to provide capacity and do not have the authority to tell any other machine what it can and cannot do.

Up until now, you have been using Docker in a single-host mode on your local machine. But Docker also can be switched into swarm mode, and that's what enables the use of swarms. Enabling **swarm mode** instantly makes the current machine a swarm manager. From then on, Docker will run the commands you execute on the swarm you're managing, rather than just on the current machine.

24.4.4.3.3 Set up your swarm

A swarm is made up of multiple nodes, which can be either physical or virtual machines. The basic concept is simple enough: run docker swarm init to enable swarm mode and make your current machine a swarm manager, then run docker swarm join on other machines to have them join the swarm as workers.

Choose a tab below to see how this plays out in various contexts. We'll use VMs to quickly create a two-machine cluster and turn it into a swarm.

24.4.4.3.4 Encore Bloqué

See also:

- <https://github.com/boot2docker/boot2docker/releases/download/v18.01.0-ce/boot2docker.iso>

```
PS C:/WINDOWS/system32> docker-machine create -d hyperv --hyperv-virtual-switch
->"myswitch" myvm1
```

```
PS C:/WINDOWS/system32> docker-machine create -d hyperv --hyperv-virtual-switch
->"myswitch" myvm1
Creating CA: C:/Users/compadm/.docker/machine/certs/ca.pem
Creating client certificate: C:/Users/compadm/.docker/machine/certs/cert.pem
Running pre-create checks...
(myvm1) Image cache directory does not exist, creating it at C:/Users/compadm/.docker/
->machine/cache...
(myvm1) No default Boot2Docker ISO found locally, downloading the latest release...
(myvm1) Latest release for github.com/boot2docker/boot2docker is v18.01.0-ce
(myvm1) Downloading C:/Users/compadm/.docker/machine/cache/boot2docker.iso from
->https://github.com/boot2docker/boot2dock
er/releases/download/v18.01.0-ce/boot2docker.iso...
Error with pre-create check: "Get https://github-production-release-asset-2e65be.s3.
->amazonaws.com/14930729/634fb5b0-f6ac-11e7-8f12-e1c4544a979b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=A
KIAIWNJYAX4CSVEH53A%2F20180115%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20180115T134730Z&X-Amz-Expires=300&X-Amz-Signature=5efdf365c94b790f1a95579a7f424a0731be82a19a2d806340d18c56085" (continues on next page)
```

(continued from previous page)

```
-SignedHeaders=host&actor_id=0&response-content-disposition=attachment%3B%20filename
↳ %3Dboot2docker.iso&response-content-type=application%2Foctet-stream: read tcp 10.0.
↳ 40.41:55806->54.231.48.184:4
43: wsarecv: Une tentative de connexion a échoué car le parti connecté n'a pas
↳ répondu convenablement au-delà d'une certaine durée ou une connexion établie a
↳ échoué car l'hôte de connexion n'a pa
s répondu."
```

Warning: impossible d'accéder au stockage Amazon S3.

Solution

Téléchargement à la maison et copie manuelle sous C:/Users/compadm/.docker/machine/cache.

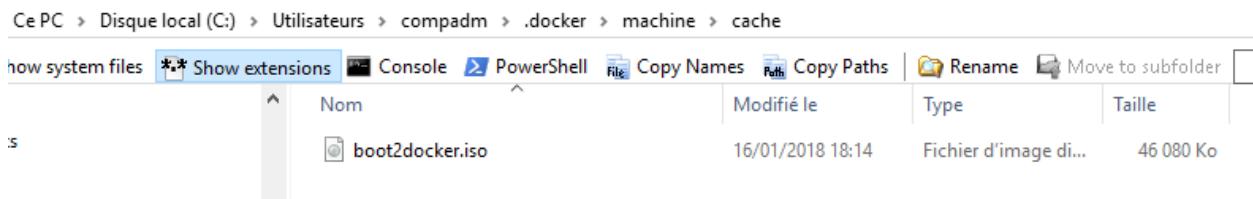


Fig. 4: <https://github.com/boot2docker/boot2docker/releases/download/v18.01.0-ce/boot2docker.iso>

24.5 A Simple Recipe for Django Development In Docker par Adam King (Advanced tutorial)

See also:

- <https://medium.com/@adamzeitcode/a-simple-recipe-for-django-development-in-docker-bonus-testing-with-selenium-6a038ec1>

Contents

- *A Simple Recipe for Django Development In Docker par Adam King (Advanced tutorial)*
 - *Dockerfile Adam King*
 - * *WORKDIR*
 - *docker-compose.yml Adam King*
 - * *stdin_open: true, tty:true*
 - * *docker-compose up -d*
 - *Explore your container (docker-compose exec django bash)*
 - *Take a break*
 - *Next Steps: Add a MySQL Database*
 - * *db*

```
* DATABASE_URL
```

adam king



Fig. 5: <https://medium.com/@adamzeitcode/a-simple-recipe-for-django-development-in-docker-bonus-testing-with-selenium-6a038ec>

24.5.1 Dockerfile Adam King

```
# My Site
# Version: 1.0

FROM python:3

# Install Python and Package Libraries
RUN apt-get update && apt-get upgrade -y && apt-get autoremove && apt-get autoclean
RUN apt-get install -y \
    libffi-dev \
    libssl-dev \
    libmysqlclient-dev \
    libxml2-dev \
    libxslt-dev \
    libjpeg-dev \
    libfreetype6-dev \
    zlib1g-dev \
    net-tools \
    vim

# Project Files and Settings
ARG PROJECT=myproject
ARG PROJECT_DIR=/var/www/${PROJECT}

RUN mkdir -p $PROJECT_DIR
WORKDIR $PROJECT_DIR
COPY Pipfile Pipfile.lock ./
RUN pip install -U pipenv
RUN pipenv install --system

# Server
EXPOSE 8000
STOPSIGNAL SIGINT
ENTRYPOINT ["python", "manage.py"]
CMD ["runserver", "0.0.0.0:8000"]
```

Without getting too deep in the weeds about creating Dockerfiles, let's take a quick look at what's going on here. We specify some packages we want installed on our Django server (The Ubuntu image is pretty bare-bones, it doesn't even come with ping!).

24.5.1.1 WORKDIR

The WORKDIR variable is interesting in this case it's setting `/var/www/myproject/` on the server as the equivalent to your Django project's root directory. We also expose port 8000 and run the server.

Note that in this case, we're using pipenv to manage our package dependencies.

24.5.2 docker-compose.yml Adam King

```
version: "2"
services:
  django:
    container_name: django_server
    build:
      context: .
      dockerfile: Dockerfile
    image: docker_tutorial_django
    stdin_open: true
    tty: true
    volumes:
      - ./var/www/myproject
    ports:
      - "8000:8000"
```

Now we can run **docker-compose build** and it'll build our image which we named `docker_tutorial_django` that will run inside a container called `django_server`.

Spin it up by running **docker-compose up**.

Before we go any further, take a quick look at that `docker-compose.yml` file. The lines,

24.5.2.1 stdin_open: true, tty:true

```
stdin_open: true
tty: true
```

are important, because they let us run an interactive terminal.

Hit `ctrl-c` to kill the server running in your terminal, and then bring it up in the background with **docker-compose up -d**

docker ps tells us it's still running.

24.5.2.2 docker-compose up -d

We need to attach to that running container, in order to see its server output and pdb breakpoints. The command `docker attach django_server` will present you with a blank line, but if you refresh your web browser, you'll see the server output.

Drop:

```
import pdb; pdb.set_trace()
```

in your code and you'll get the interactive debugger, just like you're used to.

24.5.3 Explore your container (`docker-compose exec django bash`)

With your container running, you can run the command:

```
docker-compose exec django bash
```

which is a shorthand for the command:

```
docker exec -it django_server bash.
```

You'll be dropped into a bash terminal inside your running container, with a working directory of `/var/www/myproject`, just like you specified in your Docker configuration.

This console is where you'll want to run your `manage.py` tasks: execute tests, make and apply migrations, use the python shell, etc.

24.5.4 Take a break

Before we go further, let's stop and think about what we've accomplished so far.

We've now got our Django server running in a reproducible Docker container.

If you have collaborators on your project or just want to do development work on another computer, all you need to get up and running is a copy of your:

- **Dockerfile**
- **docker-compose.yml**
- **Pipfile**

You can rest easy knowing that the environments will be identical.

When it comes time to push your code to a staging or production environment, you can build on your existing Dockerfile maybe add some error logging, a production-quality web server, etc.

24.5.5 Next Steps: Add a MySQL Database

Now, we could stop here and we'd still be in a pretty good spot, but there's still a lot of Docker goodness left on the table.

Let's add a real database.

Open up your `docker-compose.yml` file and update it:

```
version: "2"
services:
  django:
    container_name: django_server
    build:
      context: .
      dockerfile: Dockerfile
    image: docker_tutorial_django
    stdin_open: true
    tty: true
    volumes:
      - ./var/www/myproject
    ports:
```

(continues on next page)

(continued from previous page)

```

  - "8000:8000"
links:
  - db
environment:
  - DATABASE_URL=mysql://root:itsasecret@db:3306/docker_tutorial_django_db

db:
  container_name: mysql_database
  image: mysql/mysql-server
  ports:
    - "3306:3306"
  environment:
    - MYSQL_ROOT_PASSWORD=itsasecret
  volumes:
    - /Users/Adam/Development/data/mysql:/var/lib/mysql

```

24.5.5.1 db

We added a new service to our docker-compose.yml called **db**.

I named the container **mysql_database**, and we are basing it off the image mysql/mysql-server. Check out <http://hub.docker.com> for, like, a million Docker images.

24.5.5.1.1 MYSQL_ROOT_PASSWORD

We set the root password for the MySQL server, as well as expose a port (host-port:container-port) to the ‘outer world.’ We also need to specify the location of our MySQL files. I’m putting them in a directory called data in my Development directory.

In our django service, I added a link to the db service. docker-compose acts as a sort of ‘internal DNS’ for our Docker containers. If I run docker-compose up -d and then jump into my running Django container with docker-compose exec django bash, I can ping db and confirm the connection:

```
root@e94891041716:/var/www/myproject# ping db
```

```

PING db (172.23.0.3): 56 data bytes
64 bytes from 172.23.0.3: icmp_seq=0 ttl=64 time=0.232 ms
64 bytes from 172.23.0.3: icmp_seq=1 ttl=64 time=0.229 ms
64 bytes from 172.23.0.3: icmp_seq=2 ttl=64 time=0.247 ms
64 bytes from 172.23.0.3: icmp_seq=3 ttl=64 time=0.321 ms
64 bytes from 172.23.0.3: icmp_seq=4 ttl=64 time=0.310 ms
^C--- db ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.229/0.268/0.321/0.040 ms
root@e94891041716:/var/www/myproject#

```

24.5.5.2 DATABASE_URL

Adding the environment variable, `DATABASE_URL=mysql://root:itsasecret@db:3306/docker_tutorial_django_db` Will allow our Django database to use a real, production-ready version of MySQL instead of the default SQLite.

Note that you’ll need to use a package like `getenv` in your `settings.py` to read environment variables:

```
DATABASE_URL=env('DATABASE_URL')
```

If it's your first time running a MySQL server, you might have a little bit of housekeeping: setting the root password, granting privileges, etc.

Check the corresponding documentation for the server you're running. You can jump into the running MySQL server the same way:

```
$ docker-compose exec db bash
```

```
$ mysql -p itsasecret
> CREATE DATABASE docker_tutorial_django_db;
etc, etc
```

24.6 Modern DevOps with Django par Jacob Cook (Advanced tutorial)

See also:

- <https://peakwinter.net/>
- <https://twitter.com/peakwinter>
- <https://github.com/peakwinter/modern-devops-django-sample>
- <https://peakwinter.net/blog/modern-devops-django/>

Contents

- *Modern DevOps with Django par Jacob Cook (Advanced tutorial)*
 - *tree*
 - *Dockerfile Jacob Cook*
 - *docker-compose.yml Jacob Cook*
 - *Testing and Production*
 - * *docker-compose.test.yml*
 - * *docker-compose.staging.yml*
 - * *docker-compose.prod.yml*

24.6.1 tree

```
pvergaine@uc026:/mnt/y/projects_id3/P5N001/XLOGCA135_tutorial_docker/tutorial_docker/
└─tutoriels/modern_devops$ tree
```

```
└── modern-devops-django-sample
    ├── docker-compose.ci.yml
    ├── docker-compose.prod.yml
    └── docker-compose.staging.yml
```

(continues on next page)

(continued from previous page)

```

├── docker-compose.test.yml
├── docker-compose.yml
├── Dockerfile
├── LICENSE
└── manage.py
    └── modern_devops
        ├── __init__.py
        ├── settings.py
        ├── urls.py
        └── wsgi.py
    └── myapp
        ├── admin.py
        ├── apps.py
        ├── __init__.py
        ├── migrations
        │   └── __init__.py
        ├── models.py
        ├── tests.py
        └── views.py
    └── README.md
    └── requirements.txt
    └── uwsgi.ini
└── modern_devops.rst

```

24.6.2 Dockerfile Jacob Cook

```

FROM python:3-alpine3.6

ENV PYTHONUNBUFFERED=1

RUN apk add --no-cache linux-headers bash gcc \
    musl-dev libjpeg-turbo-dev libpng libpq \
    postgresql-dev uwsgi uwsgi-python3 git \
    zlib-dev libmagic

WORKDIR /site
COPY ./ /site
RUN pip install -U -r /site/requirements.txt
CMD python manage.py migrate && uwsgi --ini=/site/uwsgi.ini

```

First things first is our Dockerfile. This is the configuration that takes a base image (in our case Python 3.6 installed on a thin copy of Alpine Linux) and installs everything our application needs to run, including our Python dependencies.

It also sets a default command to use - this is the command that will be executed each time our container starts up in production.

We want it to check for any pending migrations, run them, then start up our uWSGI server to make our application available to the Internet. It's safe to do this because if any migrations failed after our automatic deployments to staging, we would be able to recover from that and make the necessary changes before we tag a release and deploy to production.

This Dockerfile example builds a container with necessary dependencies for things like image uploads as well as connections to a PostgreSQL database.

24.6.3 docker-compose.yml Jacob Cook

We can now build our application with docker build -t myapp . and run it with docker run -it myapp. But in the case of our development environment, **we are going to use Docker Compose in practice.**

The Docker Compose configuration below is sufficient for our development environment, and will serve as a base for our configurations in staging and production, which can include things like Celery workers and monitoring services.

```
version: '3'

services:
  app:
    build: ./  
    command: bash -c "python3 manage.py migrate && python3 manage.py runserver 0.0.0.  
    ↪0:8000"  
    volumes:  
      - ./site:/site:rw  
    depends_on:  
      - postgresql  
      - redis  
    environment:  
      DJANGO_SETTINGS_MODULE: myapp.settings.dev  
    ports:  
      - "8000:8000"

  postgresql:  
    restart: always  
    image: postgres:10-alpine  
    volumes:  
      - ./dbdata:/var/lib/postgresql:rw  
    environment:  
      POSTGRES_USER: myapp  
      POSTGRES_PASSWORD: myapp  
      POSTGRES_DB: myapp

  redis:  
    restart: always  
    image: redis:latest
```

This is a pretty basic configuration - all we are doing is setting a startup command for our app (similar to the entrypoint in our Docker container, except this time we are going to run Django's internal dev server instead) and initializing PostgreSQL and Redis containers that will be linked with it.

It's important to note that volumes line in our app service — this is going to bind the current directory of source code on our host machine to the installation folder inside the container.

That way we can make changes to the code locally and still use the automatic reloading feature of the Django dev server.

At this point, all we need to do is **docker-compose up**, and our Django application will be listening on port 8000, just as if we were running it from a virtualenv locally. This configuration is perfectly suitable for developer environments — all anyone needs to do to get started using the exact same environment as you is to clone the Git repository and run docker-compose up !

24.6.4 Testing and Production

For testing your application, whether that's on your local machine or via Gitlab CI, I've found it's helpful to create a clone of this docker-compose.yml configuration and customize the command directive to instead run whatever starts

your test suite. In my case, I use the Python coverage library, so I have a second file called docker-compose.test.yml which is exactly the same as the first, save for the command directive has been changed to:

```
command: bash -c "coverage run --source='.' manage.py test myapp && coverage report"
```

24.6.4.1 docker-compose.test.yml

```
version: '3'

services:
  app:
    build: .
    command: bash -c "coverage run --source='.' manage.py test kanban && coverage report"
    volumes:
      - ./site:rw
    depends_on:
      - postgresql
      - redis
    environment:
      DJANGO_SETTINGS_MODULE: modern_devops.settings.test

  postgresql:
    restart: always
    image: postgres:10-alpine
    environment:
      POSTGRES_USER: myapp_test
      POSTGRES_PASSWORD: myapp_test
      POSTGRES_DB: myapp_test

  redis:
    restart: always
    image: redis:latest
```

Then, I run my test suite locally with:

```
docker-compose -p test -f docker-compose.test.yml up.
```

24.6.4.2 docker-compose.staging.yml

```
version: '3'

services:
  app:
    image: registry.gitlab.com/path/to/myapp:staging
    environment:
      DJANGO_SETTINGS_MODULE: modern_devops.settings.staging
    volumes:
      - /var/data/myapp/staging/settings.py:/site/modern_devops/settings/staging.
    depends_on:
      - postgresql
      - redis
    networks:
```

(continues on next page)

(continued from previous page)

```

      - default
      - public

postgresql:
    image: postgres:10-alpine
    volumes:
      - /var/data/realtime/myapp/staging/db:/var/lib/postgresql/data:rw
    environment:
      POSTGRES_USER: myapp_staging
      POSTGRES_PASSWORD: myapp_staging
      POSTGRES_DB: myapp_staging

redis:
    image: redis:latest

networks:
  public:
    external: true

```

24.6.4.3 docker-compose.prod.yml

For production and staging environments, I do the same thing — duplicate the file with the few changes I need to make for the environment in particular. In this case, for production, I don't want to provide a build path — I want to tell Docker that it needs to take my application from the container registry each time it starts up.

To do so, remove the build directive and add an image one like so:

```

image: registry.gitlab.com/path/to/myapp:prod

version: '3'

services:
  app:
    image: registry.gitlab.com/path/to/myapp:prod
    environment:
      DJANGO_SETTINGS_MODULE: modern_devops.settings.prod
    volumes:
      - /var/data/myapp/prod/settings.py:/site/modern_devops/settings/prod.py:ro
    depends_on:
      - postgresql
      - redis
  networks:
    - default
    - public

postgresql:
    image: postgres:10-alpine
    volumes:
      - /var/data/realtime/myapp/prod/db:/var/lib/postgresql/data:rw
    environment:
      POSTGRES_USER: myapp_staging
      POSTGRES_PASSWORD: myapp_staging
      POSTGRES_DB: myapp_staging

redis:

```

(continues on next page)

(continued from previous page)

```
image: redis:latest

networks:
  public:
    external: true
```

24.7 Django for beginners par William Vincent

Contents

- *Django for beginners par William Vincent*
 - *Thanks to William Vincent !*
 - *tree ch4-message-board-app*
 - *Installing django with pipenv and python 3.6*
 - * *Dockerfile*
 - * *Pipfile*
 - *docker build -tag gdevops/django36_ch4 .*
 - *docker images*
 - *mb_project/settings.py*
 - *Launch the db and web services with docker-compose.yml*
 - * *db*
 - * *web*
 - * *volumes*
 - * *ports*
 - * *volumes*
 - *docker-compose run web python /code/manage.py migrate --noinput*
 - *docker-compose run web python /code/manage.py createsuperuser*
 - *docker-compose up*
 - *docker-compose ps*
 - *docker-compose exec db bash*
 - *psql -d db -U postgres*
 - * *dt*
 - * *conninfo*
 - * *dn*
 - * *d posts_post*

24.7.1 Thanks to William Vincent !

See also:

- <https://twitter.com/wsv3000>
- <https://wsvincent.com/django-docker-postgresql/>
- <https://github.com/wsvincent/djangox>
- <https://gitlab.com/gdevops/wsdjangoforbeginners>



Fig. 6: <https://twitter.com/wsv3000>

```
total 52
drwxrwxr-x. 6 pvergain pvergain 4096 28 mai  16:10 ch10-bootstrap
drwxrwxr-x. 6 pvergain pvergain 4096 28 mai  16:10 ch11-password-change-reset
drwxrwxr-x. 6 pvergain pvergain 4096 28 mai  16:10 ch12-email
```

(continues on next page)

(continued from previous page)

drwxrwxr-x. 7 pvergain pvergain 4096 28 mai	16:10 ch13-newspaper-app
drwxrwxr-x. 7 pvergain pvergain 4096 28 mai	16:10 ch14-permissions-and-
↳ authorizations	
drwxrwxr-x. 7 pvergain pvergain 4096 28 mai	16:10 ch15-comments
drwxrwxr-x. 4 pvergain pvergain 92 28 mai	16:10 ch2-hello-world-app
drwxrwxr-x. 5 pvergain pvergain 103 28 mai	16:10 ch3-pages-app
drwxrwxr-x. 5 pvergain pvergain 4096 28 mai	16:15 ch4-message-board-app
drwxrwxr-x. 7 pvergain pvergain 4096 28 mai	16:10 ch5-blog-app
drwxrwxr-x. 7 pvergain pvergain 4096 28 mai	16:10 ch6-blog-app-with-forms
drwxrwxr-x. 7 pvergain pvergain 4096 28 mai	16:10 ch7-blog-app-with-users
drwxrwxr-x. 4 pvergain pvergain 4096 28 mai	16:10 ch8-custom-user-model
drwxrwxr-x. 5 pvergain pvergain 4096 28 mai	16:10 ch9-user-authentication
-rw-rw-r--. 1 pvergain pvergain 689 28 mai	16:15 Readme.md

24.7.2 tree ch4-message-board-app

See also:

- <https://gitlab.com/gdevops/wsdjangoforbeginners>

```
tree ch4-message-board-app
```

```
ch4-message-board-app/
├── Dockerfile
├── manage.py
└── mb_project
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
├── Pipfile
├── Pipfile.lock
└── posts
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    │   └── 0001_initial.py
    │       └── __init__.py
    ├── models.py
    ├── tests.py
    ├── urls.py
    └── views.py
└── Procfile
└── templates
    └── home.html
```

24.7.3 Installing django with pipenv and python 3.6

See also:

- <https://wsvincent.com/django-docker-postgresql/>
- <https://gitlab.com/gdevops/wsdjangoforbeginners/blob/master/ch4-message-board-app/Dockerfile>

24.7.3.1 Dockerfile

```
cat Dockerfile

FROM python:3.6

ENV PYTHONUNBUFFERED 1

COPY . /code/
WORKDIR /code/

RUN pip install pipenv
RUN pipenv install --system

EXPOSE 8000
```

24.7.3.2 Pipfile

```
$ cat Pipfile

[[source]]
url = "https://pypi.python.org/simple"
verify_ssl = true
name = "pypi"

[packages]
gunicorn = "*"
django = "*"
"psycopg2" = "*"

[dev-packages]

[requires]
python_version = "3.6"
```

24.7.4 docker build –tag gdevops/django36_ch4 .

See also:

- *docker build*

We can't run a Docker container until it has an image so let's do that by *building it*.

```
docker build --tag gdevops/django36_ch4 .
```

```
Sending build context to Docker daemon 48.82MB
Step 1/7 : FROM python:3.6
3.6: Pulling from library/python
cc1a78bfd46b: Pull complete
6861473222a6: Pull complete
7e0b9c3b5ae0: Pull complete
3ec98735f56f: Pull complete
9b311b87a021: Pull complete
```

(continues on next page)

(continued from previous page)

```

048165938570: Pull complete
1ca3d78efb22: Pull complete
0f6c8999c3b7: Pull complete
5a85410f5000: Pull complete
Digest: sha256:52a2bd143faf6430b182b56a5fdeb70f26b8ca8fdbd40210c3ed8a8ee1eaba343
Status: Downloaded newer image for python:3.6
--> 29d2f3226daf
Step 2/7 : ENV PYTHONUNBUFFERED 1
--> Running in eelad676c613
Removing intermediate container eelad676c613
--> 8fb46520978f
Step 3/7 : COPY . /code/
--> 3a6b8395d66b
Step 4/7 : WORKDIR /code/
Removing intermediate container 72885dc811e1
--> e0c26850ab99
Step 5/7 : RUN pip install pipenv
--> Running in 19666f95212b
Collecting pipenv
  Downloading https://files.pythonhosted.org/packages/8d/fe/
-> 4faa519acfb984015dde4e2973d89e47baf6a9cd81e9d58e2f2d3c47fb6f/pipenv-2018.5.18-py3-
-> none-any.whl (6.4MB)
Requirement already satisfied: setuptools>=36.2.1 in /usr/local/lib/python3.6/site-
-> packages (from pipenv) (39.1.0)
Collecting certifi (from pipenv)
  Downloading https://files.pythonhosted.org/packages/7c/e6/
-> 92ad559b7192d846975fc916b65f667c7b8c3a32bea7372340bfe9a15fa5/certifi-2018.4.16-py2.
-> py3-none-any.whl (150kB)
Collecting virtualenv (from pipenv)
  Downloading https://files.pythonhosted.org/packages/b6/30/
-> 96a02b2287098b23b875bc8c2f58071c35d2efe84f747b64d523721dc2b5/virtualenv-16.0.0-py2.
-> py3-none-any.whl (1.9MB)
Collecting virtualenv-clone>=0.2.5 (from pipenv)
  Downloading https://files.pythonhosted.org/packages/6d/c2/
-> dccb5ccf599e0c5d1eea6acbd058af7a71384f9740179db67a9182a24798/virtualenv_clone-0.3.0-
-> py2.py3-none-any.whl
Requirement already satisfied: pip>=9.0.1 in /usr/local/lib/python3.6/site-packages_
-> (from pipenv) (10.0.1)
Installing collected packages: certifi, virtualenv, virtualenv-clone, pipenv
Successfully installed certifi-2018.4.16 pipenv-2018.5.18 virtualenv-16.0.0_
-> virtualenv-clone-0.3.0
Removing intermediate container 19666f95212b
--> 2f8d9ee873ca
Step 6/7 : RUN pipenv install --system
--> Running in baca593927a8
Installing dependencies from Pipfile.lock (c2c6d4)...
Removing intermediate container baca593927a8
--> 2d402a8f0e26
Step 7/7 : EXPOSE 8000
--> Running in c3e7a4b032d8
Removing intermediate container c3e7a4b032d8
--> b44a8c214cdf
Successfully built b44a8c214cdf
Successfully tagged gdevops/django36_ch4:latest

```

24.7.5 docker images

See also:

- *docker images*

```
$ docker images --no-trunc
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
gdevops/django36_ch4	latest	sha256:b44a8c214cdfe23f6f8a4481277028fd56b1adcce615f92f703ddca728e054e0	5 minutes	
python	3.6	sha256:29d2f3226daf297b27f0240244f4e8d614fb63eeab8cd09d816e8b7b04d1c011	3 weeks	
		ago	911MB	

24.7.6 mb_project/settings.py

```
# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

# https://djangoforbeginners.com/docker-postgresql/
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'HOST': 'db', # set in docker-compose.yml
        'PORT': 5432 # default postgres port
    }
}
```

24.7.7 Launch the db and web services with docker-compose.yml

```
version: '3'

services:
  db:
    image: postgres:10.4
    volumes:
      - postgres_data:/var/lib/postgresl/data/

  web:
    build: .
    command: python /code/manage.py migrate --noinput
    command: python /code/manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db
```

(continues on next page)

(continued from previous page)

```
volumes:
  postgres_data:
```

On the top line we're using the most recent version of Compose which is **3**.

24.7.7.1 db

Under **db** for the database we want the Docker image for Postgres 10.4 and use volumes to tell Compose where the container should be located in our Docker container.

24.7.7.2 web

For **web** we're specifying how the web service will run. First Compose needs to build an image from the current directory, automatically run migrations and hide the output, then start up the server at 0.0.0.0:8000.

24.7.7.3 volumes

We use volumes to tell Compose to store the code in our Docker container at /code/.

Warning: Cela nous permet d'avoir accès à notre code sur le host.

24.7.7.4 ports

The ports config lets us map our own port 8000 to the port 8000 in the Docker container.

And finally depends_on says that we should start the db first before running our web services.

24.7.7.5 volumes

The last section volumes is because Compose has a rule that you must list named volumes in a top-level volumes key. Docker is all set!

24.7.8 docker-compose run web python /code/manage.py migrate --noinput

```
$ docker-compose run web python /code/manage.py migrate --noinput
```

```
Creating network "ch4-message-board-app_default" with the default driver
Creating volume "ch4-message-board-app_postgres_data" with default driver
Pulling db (postgres:10.4)...
10.4: Pulling from library/postgres
f2aa67a397c4: Pull complete
8218dd41bf94: Pull complete
e9b7fa2e6bd8: Pull complete
7288a45ee17f: Pull complete
0d0f8a67376c: Pull complete
972b115243de: Pull complete
d38528c83dd1: Pull complete
```

(continues on next page)

(continued from previous page)

```
9be166d23dee: Pull complete
12015b5ceae7: Pull complete
363876c09ce9: Pull complete
b810ba8b2ac0: Pull complete
e1ee11d636cf: Pull complete
50d32813cba1: Pull complete
4f0109485c03: Pull complete
Digest: sha256:1acf72239c685322579be2116dc54f8a25fc4523882df35171229c9fee3b3b17
Status: Downloaded newer image for postgres:10.4
Creating ch4-message-board-app_db_1 ... done
Building web
Step 1/7 : FROM python:3.6
--> 29d2f3226daf
Step 2/7 : ENV PYTHONUNBUFFERED 1
--> Using cache
--> 8fb46520978f
Step 3/7 : COPY . /code/
--> 3b31f2bb6016
Step 4/7 : WORKDIR /code/
Removing intermediate container 183aa302c2d1
--> fd032580fe90
Step 5/7 : RUN pip install pipenv
--> Running in 8f333f7716ee
Collecting pipenv
  Downloading https://files.pythonhosted.org/packages/8d/fe/
  ↳ 4faa519acfb984015dde4e2973d89e47baf6a9cd81e9d58e2f2d3c47fb6f/pipenv-2018.5.18-py3-
  ↳ none-any.whl (6.4MB)
Collecting certifi (from pipenv)
  Downloading https://files.pythonhosted.org/packages/7c/e6/
  ↳ 92ad559b7192d846975fc916b65f667c7b8c3a32bea7372340bfe9a15fa5/certifi-2018.4.16-py2.
  ↳ py3-none-any.whl (150kB)
Requirement already satisfied: setuptools>=36.2.1 in /usr/local/lib/python3.6/site-
  ↳ packages (from pipenv) (39.1.0)
Requirement already satisfied: pip>=9.0.1 in /usr/local/lib/python3.6/site-packages
  ↳ (from pipenv) (10.0.1)
Collecting virtualenv (from pipenv)
  Downloading https://files.pythonhosted.org/packages/b6/30/
  ↳ 96a02b2287098b23b875bc8c2f58071c35d2efe84f747b64d523721dc2b5/virtualenv-16.0.0-py2.
  ↳ py3-none-any.whl (1.9MB)
Collecting virtualenv-clone>=0.2.5 (from pipenv)
  Downloading https://files.pythonhosted.org/packages/6d/c2/
  ↳ dccb5ccf599e0c5d1eea6acbd058af7a71384f9740179db67a9182a24798/virtualenv_clone-0.3.0-
  ↳ py2.py3-none-any.whl
Installing collected packages: certifi, virtualenv, virtualenv-clone, pipenv
Successfully installed certifi-2018.4.16 pipenv-2018.5.18 virtualenv-16.0.0
  ↳ virtualenv-clone-0.3.0
Removing intermediate container 8f333f7716ee
--> f27a4a1e4257
Step 6/7 : RUN pipenv install --system
--> Running in 2519166487e4
Installing dependencies from Pipfile.lock (c2c6d4)...
Removing intermediate container 2519166487e4
--> 255cb3b345c2
Step 7/7 : EXPOSE 8000
--> Running in 8cb1c964976e
Removing intermediate container 8cb1c964976e
--> 376512737492
```

(continues on next page)

(continued from previous page)

```
Successfully built 376512737492
Successfully tagged ch4-message-board-app_web:latest
WARNING: Image for service web was built because it did not already exist. To rebuild ↵
this image you must use `docker-compose build` or `docker-compose up --build`.
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, posts, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying posts.0001_initial... OK
  Applying sessions.0001_initial... OK
```

24.7.9 docker-compose run web python /code/manage.py createsuperuser

```
$ docker-compose run web python /code/manage.py createsuperuser
```

WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All ↵
containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

```
Starting ch4messageboardapp_db_1 ... done
Username (leave blank to use 'root'):
Email address: patrick.vergain@id3.eu
Password:
Password (again):
The password is too similar to the email address.
This password is too short. It must contain at least 8 characters.
Password:
Password (again):
Superuser created successfully.
```

24.7.10 docker-compose up

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_ ↵
docker\tutoriels\djangoforbeginners\ch4-message-board-app>docker-compose up
```

WARNING: The Docker Engine you're using is running in swarm mode.

(continues on next page)

```
3\PSN001\XLOGCA135_tutorial_docker\tutoriels\djangoforbeginners\ch4-message-board-app>docker-compose run web python /code/manage.py createsuperuser
Docker Engine you're using is running in swarm mode.
You can use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.
To use swarm mode across the swarm, use `docker stack deploy`.

ssageboardapp_db_1 ... done
e blank to use 'root'):
patrick.vergain@id3.eu

n):
s too similar to the email address,
is too short. It must contain at least 8 characters.

n):
ted successfully.

3\PSN001\XLOGCA135_tutorial_docker\tutoriels\djangoforbeginners\ch4-message-board-app>
```

Fig. 7: docker-compose run web python /code/manage.py createsuperuser

(continued from previous page)

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

```
ch4messageboardapp_db_1 is up-to-date
Creating ch4messageboardapp_web_1 ... done
Attaching to ch4messageboardapp_db_1, ch4messageboardapp_web_1
db_1    | The files belonging to this database system will be owned by user "postgres".
db_1    | This user must also own the server process.
db_1    |
db_1    | The database cluster will be initialized with locale "en_US.utf8".
db_1    | The default database encoding has accordingly been set to "UTF8".
db_1    | The default text search configuration will be set to "english".
db_1    |
db_1    | Data page checksums are disabled.
db_1    |
db_1    | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db_1    | creating subdirectories ... ok
db_1    | selecting default max_connections ... 100
db_1    | selecting default shared_buffers ... 128MB
db_1    | selecting dynamic shared memory implementation ... posix
db_1    | creating configuration files ... ok
db_1    | running bootstrap script ... ok
db_1    | performing post-bootstrap initialization ... ok
db_1    | syncing data to disk ... ok
db_1    |
db_1    | Success. You can now start the database server using:
db_1    |
db_1    |     pg_ctl -D /var/lib/postgresql/data -l logfile start
db_1    |
db_1    |
db_1    | WARNING: enabling "trust" authentication for local connections
db_1    | You can change this by editing pg_hba.conf or using the option -A, or
db_1    | --auth-local and --auth-host, the next time you run initdb.
db_1    | ****
db_1    | WARNING: No password has been set for the database.
db_1    |           This will allow anyone with access to the
db_1    |           Postgres port to access your database. In
db_1    |           Docker's default configuration, this is
db_1    |           effectively any other container on the same
```

(continues on next page)

(continued from previous page)

```

db_1 |           system.
db_1 |
db_1 |           Use "-e POSTGRES_PASSWORD=password" to set
db_1 |           it in "docker run".
db_1 | ****
db_1 | waiting for server to start....2018-01-23 08:34:30.556 UTC [39] LOG: ↵
db_1 |   ↵listening on IPv4 address "127.0.0.1", port 5432
db_1 | 2018-01-23 08:34:30.557 UTC [39] LOG:  could not bind IPv6 address "::1": ↵
db_1 |   ↵Cannot assign requested address
db_1 | 2018-01-23 08:34:30.557 UTC [39] HINT:  Is another postmaster already ↵
db_1 |   ↵running on port 5432? If not, wait a few seconds and retry.
db_1 | 2018-01-23 08:34:30.682 UTC [39] LOG:  listening on Unix socket "/var/run/ ↵
db_1 |   ↵postgresql/.s.PGSQL.5432"
db_1 | 2018-01-23 08:34:30.865 UTC [40] LOG:  database system was shut down at 2018- ↵
db_1 |   ↵01-23 08:34:28 UTC
db_1 | 2018-01-23 08:34:30.928 UTC [39] LOG:  database system is ready to accept ↵
db_1 |   ↵connections
db_1 |   done
db_1 |   server started
db_1 |   ALTER ROLE
db_1 |
db_1 |
db_1 |   /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
db_1 |
db_1 | 2018-01-23 08:34:31.493 UTC [39] LOG:  received fast shutdown request
db_1 | waiting for server to shut down....2018-01-23 08:34:31.557 UTC [39] LOG: ↵
db_1 |   ↵aborting any active transactions
db_1 | 2018-01-23 08:34:31.559 UTC [39] LOG:  worker process: logical replication ↵
db_1 |   ↵launcher (PID 46) exited with exit code 1
db_1 | 2018-01-23 08:34:31.560 UTC [41] LOG:  shutting down
db_1 | 2018-01-23 08:34:32.052 UTC [39] LOG:  database system is shut down
db_1 |   done
db_1 |   server stopped
db_1 |
db_1 | PostgreSQL init process complete; ready for start up.
db_1 |
db_1 | 2018-01-23 08:34:32.156 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", ↵
db_1 |   ↵port 5432
db_1 | 2018-01-23 08:34:32.156 UTC [1] LOG:  listening on IPv6 address "::", port ↵
db_1 |   ↵5432
db_1 | 2018-01-23 08:34:32.256 UTC [1] LOG:  listening on Unix socket "/var/run/ ↵
db_1 |   ↵postgresql/.s.PGSQL.5432"
db_1 | 2018-01-23 08:34:32.429 UTC [57] LOG:  database system was shut down at 2018- ↵
db_1 |   ↵01-23 08:34:31 UTC
db_1 | 2018-01-23 08:34:32.483 UTC [1] LOG:  database system is ready to accept ↵
db_1 |   ↵connections
web_1 | Performing system checks...
web_1 |
web_1 | System check identified no issues (0 silenced).
web_1 | January 23, 2018 - 08:46:09
web_1 | Django version 2.0.1, using settings 'mb_project.settings'
web_1 | Starting development server at http://0.0.0.0:8000/
web_1 | Quit the server with CONTROL-C.

```

We can confirm it works by navigating to <http://127.0.0.1:8000/> where you'll see the same homepage as before.

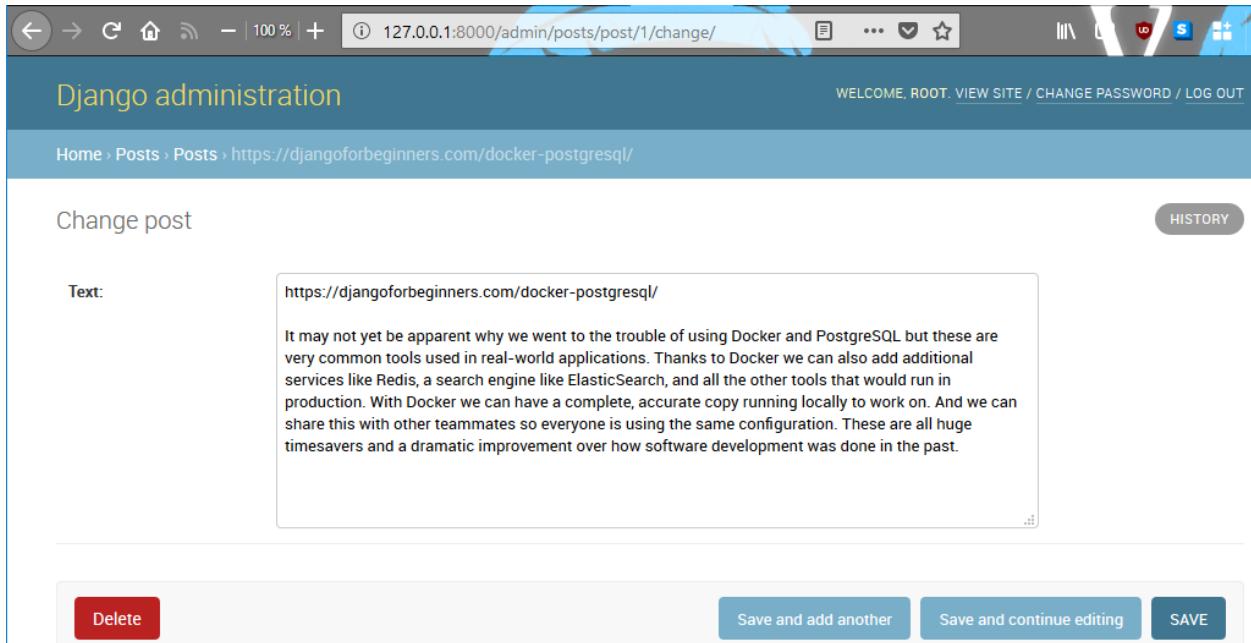


Fig. 8: `http://127.0.0.1:8000/admin/posts/post/1/change/`

24.7.11 docker-compose ps

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\william_<vincent\ch4-message-board-app> docker-compose ps
```

Name	Command	State	Ports
<hr/>			
ch4messageboardapp_db_1	docker-entrypoint.sh postgres	Up	5432/tcp
ch4messageboardapp_web_1	python /code/manage.py run ...	Up	0.0.0.0:8000-> 8000/tcp
<hr/>			

24.7.12 docker-compose exec db bash

```
docker-compose exec db bash
```

24.7.13 psql -d db -U postgres

```
root@ee941cf5bc20:/# psql -U postgres
```

```
psql (10.1)
Type "help" for help.
```

24.7.13.1 dt

```
postgres=# \dt
```

List of relations			
Schema	Name	Type	Owner
public	auth_group	table	postgres
public	auth_group_permissions	table	postgres
public	auth_permission	table	postgres
public	auth_user	table	postgres
public	auth_user_groups	table	postgres
public	auth_user_user_permissions	table	postgres
public	django_admin_log	table	postgres
public	django_content_type	table	postgres
public	django_migrations	table	postgres
public	django_session	table	postgres
public	posts_post	table	postgres

(11 rows)

24.7.13.2 conninfo

```
postgres=# \conninfo
```

```
You are connected to database "postgres" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
```

```
postgres=# \l
```

List of databases						
Name	Owner	Encoding	Collate	Ctype	Access privileges	
postgres	postgres	UTF8	en_US.utf8	en_US.utf8		
template0	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres	+
					postgres=CTc/postgres	
template1	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres	+
					postgres=CTc/postgres	

(3 rows)

24.7.13.3 dn

```
postgres=# \dn
List of schemas
 Name | Owner
-----+-----
 public | postgres
(1 row)
```

24.7.13.4 d posts_post

```
postgres=# \d posts_post
```

```
Table "public.posts_post"
Column | Type | Collation | Nullable | Default
-----+-----+-----+-----+
id | integer | | not null | nextval('posts_post_id_seq'::regclass)
text | text | | not null |
Indexes:
"posts_post_pkey" PRIMARY KEY, btree (id)
```

24.8 A Brief Intro to Docker for Djangonauts par Lacey Williams

See also:

- <https://twitter.com/laceynwilliams>
- <https://twitter.com/laceynwilliams/status/921421761039818754>
- <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>
- <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>
- <https://www.revsys.com/tidbits/docker-useful-command-line-stuff/>
- <https://www.youtube.com/watch?v=v5jfDDg55xs&feature=youtu.be&a=>

Contents

- *A Brief Intro to Docker for Djangonauts par Lacey Williams*
 - *Introduction*
 - *Annonce de l'écriture du tutoriel le 20 octobre 2017*
 - *Dockerfile Lacey Williams*
 - * *FROM python:3.6*
 - * *ENV PYTHONUNBUFFERED 1*
 - * *ENV DJANGO_ENV dev*
 - * *ENV DOCKER_CONTAINER 1*
 - * *EXPOSE 8000*
 - *docker-compose.yml Lacey Williams*
 - *version: '3'*
 - *services*
 - * *db*
 - *volumes*
 - * *web*
 - *build .*

- *command: python /code/manage.py migrate --noinput*
- *command: python /code/manage.py runserver 0.0.0.0:8000*

24.8.1 Introduction

I'll be honest: I was pretty trepidatious about using Docker.

It wasn't something we used at my last job and most tutorials felt like this comic by Van Okttop.

24.8.2 Annonce de l'écriture du tutoriel le 20 octobre 2017

24.8.3 Dockerfile Lacey Williams

```
FROM python:3.6

ENV PYTHONUNBUFFERED 1
ENV DJANGO_ENV dev
ENV DOCKER_CONTAINER 1

COPY ./requirements.txt /code/requirements.txt
RUN pip install -r /code/requirements.txt

COPY . /code/
WORKDIR /code/

EXPOSE 8000
```

24.8.3.1 FROM python:3.6

You don't need to create your Docker image from scratch. You can base your image off of code in another image in the Docker Hub, a repository of existing Docker images.

On this line, I've told Docker to base my image off of the Python 3.6 image, which (you guessed it) contains Python 3.6. Pointing to Python 3.6 versus 3.6.x ensures that we get the latest 3.6.x version, which will include bug fixes and security updates for that version of Python.

24.8.3.2 ENV PYTHONUNBUFFERED 1

ENV creates an environment variable called PYTHONUNBUFFERED and sets it to 1 (which, remember, is "truthy"). All together, this statement means that Docker won't buffer the output from your application; instead, you will get to see your output in your console the way you're used to.

24.8.3.3 ENV DJANGO_ENV dev

If you use multiple environment-based settings.py files, this creates an environment variable called DJANGO_ENV and sets it to the development environment.

You might call that "test" or "local" or something else.



A circular profile picture of a young woman with long brown hair, smiling. The background is a blurred outdoor scene with greenery and a blue sky.

Lacey Williams Henschel

@laceynwilliams

You might know me from [@djangocon](#) and [@revsys](#), or from [@djangogirls](#) [@djangogirlspdx](#) [@djangogirlsatx](#) and [@treehouse](#)

📍 Portland, OR

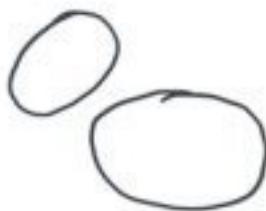
🔗 laceyhenschel.com

📅 Inscrit en mars 2009

Fig. 9: <https://twitter.com/laceynwilliams>

HOW TO: DRAW A HORSE

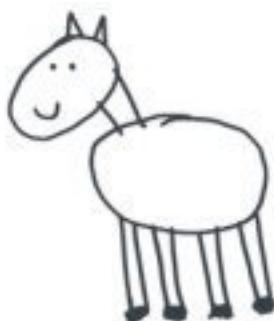
BY VAN OKTOP



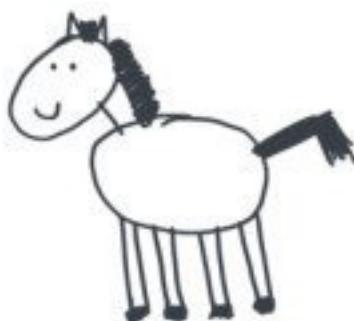
① DRAW 2 CIRCLES



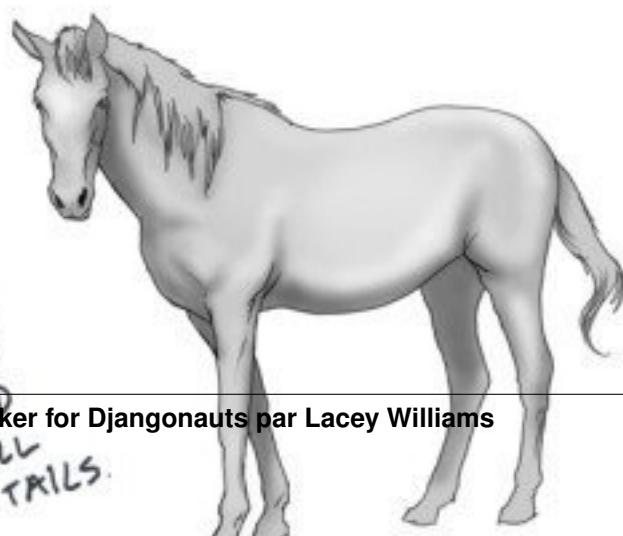
② DRAW THE LEGS



③ DRAW THE FACE



④ DRAW THE HAIR



⑤

ADD
SMALL
DETALS.



Lacey Williams Henschel

@laceynwilliams

Abonné



🐳 ICYMI: I wrote about using Docker and Django at [@revsys](#). Are you a Docker beginner?

This is for you:

🌐 À l'origine en anglais



A Brief Intro to Docker for Djangonauts

Lacey didn't have the opportunity to work with Docker at her last job. In this tidbit she steps you through getting started with Docker for Django developers.

revsys.com

19:04 - 20 oct. 2017

Fig. 11: <https://www.revsys.com/tidbits/brief-intro-docker-djangonauts/>

24.8.3.4 ENV DOCKER_CONTAINER 1

This creates an environment variable called DOCKER_CONTAINER that you can use in settings.py to load different databases depending on whether you're running your application inside a Docker container.

24.8.3.5 EXPOSE 8000

In order to runserver like a champ, your Docker container will need access to port 8000. This bestows that access.

Huzzah! Your first Dockerfile is ready to go.

24.8.4 docker-compose.yml Lacey Williams

Docker Compose lets you run more than one container in a Docker application. It's especially useful if you want to have a database, like Postgres, running in a container alongside your web app. (Docker's overview of Compose is helpful.) Compose allows you to define several services that will make up your app and run them all together.

Examples of services you might define include:

- web: defines your web service
- db: your database
- redis or another caching service

Compose can also help you relate those services to each other. For example, you likely don't want your web service to start running until your db is ready, right?

Create a new file called **docker-compose.yml** in the same directory as your Dockerfile. While Dockerfile doesn't have an extension, the docker-compose file is written in YAML, so it has the extension .yml.

Mine defines two services, web and db, and looks like this:

```
version: '3'

services:
  db:
    image: postgres:9.6.5
    volumes:
      - postgres_data:/var/lib/postgresql/data/
  web:
    build: .
    command: python /code/manage.py migrate --noinput
    command: python /code/manage.py runserver 0.0.0.0:8000
    volumes:
      - .:/code
    ports:
      - "8000:8000"
    depends_on:
      - db

volumes:
  postgres_data:
```

Just like we did with the Dockerfile, let's go through the parts of this docker-compose.yml file.

24.8.5 version: ‘3’

This line defines the version of Compose we want to use. We’re using version 3, the most recent version.

24.8.6 services

Indented under this line, we will define the services we want our image to run in separate containers when we run our project.

24.8.6.1 db

```
db:  
  image: postgres:9.6.5  
  volumes:  
    - postgres_data:/var/lib/postgresql/data/
```

This is where Compose gets exciting: this section sets up the db service as a Postgres database and instructs Compose to pull version 9.6.5 of Postgres from the image that already exists in Docker Hub. This means that I don’t need to download Postgres on my computer at all in order to use it as my local database.

Upgrading Postgres from one minor version to another while keeping your data requires running some extra scripts, pgdump and psql, and can get a little complicated. If you don’t want to mess with this, set your Postgres image to a specific version (like 9.6.5). You will probably want to upgrade the Postgres version eventually, but this will save you from having to upgrade with every minor version release.

24.8.6.1.1 volumes

volumes tells Compose where in the container I would like it to store my data: in /var/lib/postgresql/data/.

Remember when I said that each container had its own set of subdirectories and that is why you needed to copy your application code into a directory named /code/? /var/ is one of those other subdirectories.

A **volume** also lets your data persist beyond the lifecycle of a specific container.

24.8.6.2 web

```
web:  
  build: .  
  command: python /code/manage.py migrate --noinput  
  command: python /code/manage.py runserver 0.0.0.0:8000  
  volumes:  
    - ./code  
  ports:  
    - "8000:8000"  
  depends_on:  
    - db
```

This section sets up the web service, the one that will run my application code.

24.8.6.2.1 build .

build: . tells Compose to build the image from the current directory.

24.8.6.2.2 command: `python /code/manage.py migrate --noinput`

command: `python /code/manage.py migrate --noinput` will automatically run migrations when I run the container and hide the output from me in the console.

24.8.6.2.3 command: `python /code/manage.py runserver 0.0.0.0:8000`

command: `python /code/manage.py runserver 0.0.0.0:8000` will start the server when I run the container.

24.9 Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron

See also:

- <https://mbaron.developpez.com/>
- <https://twitter.com/mickaelbaron>
- <http://www.ros.org/>
- https://hub.docker.com/_/ros/
- https://github.com/osrf/docker_images
- <https://mbaron.developpez.com/tutoriels/ros/environnement-developpement-ros-docker/>
- <https://www.developpez.net/forums/d1857234/general-developpement/programmation-systeme/embarque/tutoriel-preparer-environnement-developpement-ros-docker/>



Fig. 12: Mickael Baron

Contents

- *Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron*
 - *Format PDF*
 - *Introduction*
 - *Conclusion*

24.9.1 Format PDF

24.9.2 Introduction

Ce tutoriel s'intéresse à présenter ROS (Robot Operating System) et à décrire comment proposer à un développeur un environnement de développement prêt à l'emploi quel que soit le système d'exploitation utilisé pour le développement et pour le déploiement.

24.9. Tutoriel pour préparer son environnement de développement ROS avec Docker de Mickael Baron

En effet, par défaut, ROS n'est disponible que sous un système Linux. Il existe bien des versions pour macOS, mais elles sont expérimentales et il n'existe aucune version pour Windows. Par ailleurs, même si nous souhaitons utiliser ces versions expérimentales, aurions-nous le même comportement une fois notre programme déployé ? Bien entendu, si du matériel doit être utilisé spécifiquement (bras robotisé, carte Raspberry...) pour un système d'exploitation donné (généralement sous Linux), nous n'aurions pas le choix d'utiliser le système en question.

24.9.3 Conclusion

Ce tutoriel a montré comment utiliser Docker pour le développement des applications basées sur ROS (Robot Operating System).

Plus précisément avec Docker, nous avons vu :

- comment enrichir l'image ROS ;
- comment utiliser les outils fournis par ROS (rostopic) en exécutant un conteneur ;
- comment exécuter une application ROS de plusieurs nœuds en créant plusieurs conteneurs « à la main » ou via l'outil d'orchestration docker-compose ;
- comment démarrer des nœuds qui possèdent des interfaces graphiques via le dépôt d'affichage (serveur X11) ;
- comment déployer une application ROS sur plusieurs machines physiques via l'utilisation de docker-machine ;
- comment autoriser un conteneur à accéder aux éléments matériels du système hôte ;
- comment synchroniser son répertoire de travail sur plusieurs machines.

De nombreuses perspectives sont à explorer :

- faciliter l'usage des lignes de commandes Docker via l'utilisation d'alias spécifiques. Nous pourrions aller plus loin en masquant la complexité de Docker dans un environnement de développement préconfiguré et plus simple en réduisant la longueur des lignes de commandes ;
- effectuer de la cross-compilation pour des programmes ROS développés en C++. Nous avons choisi la simplicité avec Python, mais dans le cas de C++ il y a une compilation à réaliser en amont. Qu'en est-il lorsqu'il faut compiler sur une plateforme matérielle différente X86 vs ARM ;
- utiliser Swarm de la famille Docker pour la création d'un cluster ;
- gérer le problème de la redondance de nœuds par Docker.

Dans le prochain tutoriel consacré à ROS, nous nous intéresserons à la possibilité de développer des programmes ROS avec le langage Java. Nous montrerons que ROS une architecture où plusieurs programmes développés dans des langages différents peuvent communiquer. À ce titre, nous ferons communiquer des nœuds écrits dans des langages différents tels que Python et Java.

24.10 Docker: les bons réflexes à adopter par Paul MARS (MISC 95)

See also:

- <https://www.miscmag.com/misc-n95-references-de-larticle-docker-les-bons-reflexes-a-adopter/>
- <https://www.miscmag.com/misc-n95-references-de-larticle-apercu-de-la-securite-de-docker/>

Contents

- *Docker: les bons réflexes à adopter par Paul MARS (MISC 95)*

- *Dockerfile MISC 95*
- *Fichiers .env*

24.10.1 Dockerfile MISC 95

```
FROM python:2.7-alpine # usage d'une image de base du dépôt officiel
LABEL description "Internal info on the challenge" version "0.1" # ajout
d'informations à l'image pour pouvoir l'identifier plus facilement
WORKDIR /opt/app/ # définition d'un dossier de travail pour l'exécution
des instructions suivantes
RUN addgroup -S ndh && adduser -S -g ndh ndh # exécution d'une commande
dans l'image
USER ndh
COPY requirements.txt /opt/app/ # copie de plusieurs ressources depuis
l'hôte vers l'image
COPY flag.txt /etc/x.b64
RUN pip install -r requirements.txt
RUN rm requirements.txt
COPY wsgi.py /opt/app/
COPY cmd.sh /opt/app/
COPY xml_challenge /opt/app/xml_challenge
EXPOSE 8002 # définition de la liste des ports que les conteneurs
instanciés sur l'image pourraient exposer
CMD [ "/bin/sh", "cmd.sh" ] # définition de la commande qui sera
lancée à l'instanciation d'un conteneur à partir de l'image
```

Vous aurez noté la présence d'une directive USER dans le Dockerfile précédent ainsi que de la création d'un utilisateur **ndh** quelques lignes plus haut. Par défaut, un processus lancé dans un conteneur s'exécute en tant que **root**. Vous vous doutez que ce comportement par défaut n'est pas une bonne pratique. Docker propose la directive USER permettant d'opérer le changement d'utilisateur.

Il faut simplement l'avoir créé avant dans le Dockerfile ou qu'il soit présent dans l'image sur laquelle se base la vôtre. Toutes les commandes exécutées au sein de l'image et du conteneur instancié sur cette image seront effectuées avec cet utilisateur après la directive. Pour chacun des services, il a été créé un utilisateur **ndh** dont les droits ont été modulés en fonction des besoins (besoin d'un shell ou non, droits sur certains fichiers). En pratique, cela a permis de donner un shell aux utilisateurs afin qu'ils récupèrent un drapeau sur le serveur sans qu'ils puissent le modifier ou changer l'environnement d'exécution du service.

La présence de secrets dans un Dockerfile ou un fichier docker-compose.yml.

24.10.2 Fichiers .env

Ces fichiers sont destinés à être versionnés et manipulés par plusieurs équipes. Docker dispose de fonctionnalités de gestion des secrets à travers la commande docker secrets (vous vous en doutiez, n'est-ce pas ?).

En parallèle de cette commande, une bonne pratique est de gérer les secrets par variable d'environnement et de passer ces variables à l'instanciation via la lecture d'un fichier de configuration.

24.11 Tutoriel Django step by step

See also:

- <https://blog.devartis.com/django-development-with-docker-a-step-by-step-guide-525c0d08291>

Contents

- *Tutoriel Django step by step*

24.12 Tutoriel erroneousboat Docker Django

See also:

- <https://github.com/erroneousboat/docker-django>

Contents

- *Tutoriel erroneousboat Docker Django*
 - *tree*
 - *docker-compose.yml*

24.12.1 tree

```
pvergain@uc026:/mnt/y/projects_id3/P5N001/XLOGCA135_tutorial_docker/tutorial_docker/  
↳ tutoriels/docker_django$ tree
```

```
.  
├── circle.yml  
├── config  
│   └── environment  
│       └── development.env  
├── docker-compose.yml  
├── docker_django.rst  
├── LICENSE  
├── README.md  
└── services  
    └── webserver  
        ├── config  
        │   ├── localhost.crt  
        │   ├── localhost.key  
        │   └── nginx.tpl  
        └── Dockerfile  
└── webapp  
    ├── config  
    │   ├── database-check.py  
    │   ├── django-uwsgi.ini  
    │   ├── requirements.txt  
    │   └── start.sh  
    ├── Dockerfile  
    └── starter  
        └── manage.py
```

(continues on next page)

(continued from previous page)

```

    └── starter
        ├── __init__.py
        ├── settings.py
        ├── urls.py
        └── wsgi.py

9 directories, 21 files

```

24.12.2 docker-compose.yml

```

#####
# Docker compose YAML file
#
# For documentation see: https://docs.docker.com/compose/yml/
#####

version: "3"

volumes:
  static-files:

services:
  db:
    image: postgres:10.1
    volumes:
      - /opt/starter/pgsql:/var/lib/postgresql/data/pgdata
    env_file:
      - ./config/environment/development.env

  webserver:
    build:
      context: .
      dockerfile: services/webserver/Dockerfile
    ports:
      - "80:80"
      - "443:443"
    depends_on:
      - webapp
    volumes:
      - static-files:/srv/static-files
    env_file:
      - ./config/environment/development.env

  webapp:
    build:
      context: webapp
    volumes:
      - ./webapp/starter:/srv/starter
      - static-files:/srv/static-files
    expose:
      - "8000"
    depends_on:
      - db
    env_file:

```

(continues on next page)

(continued from previous page)

```
- ./config/environment/development.env
```

24.13 Tutoriel Utilisation de pipenv avec Docker

See also:

- <https://github.com/dfederschmidt/docker-pipenv-sample>
- <https://github.com/pypa/pipenv/blob/master/Dockerfile>

Contents

- *Tutoriel Utilisation de pipenv avec Docker*
 - *Les fichiers*
 - *Réécriture du fichier Dockerfile*
 - *app.py*
 - *docker build -t docker-pipenv-sample . : construction de l'image*
 - *docker run -p 5000:5000 docker-pipenv-sample*
 - *http://localhost:5000/*
 - *docker ps*
 - *docker exec -it 1a0a3dc7924d bash*
 - *docker rm 1a0a3dc7924d: suppression du conteneur à l'arrêt*
 - *docker rmi docker-pipenv-sample: suppression de l'image*

24.13.1 Les fichiers

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\pipenv>dir
```

Le volume dans le lecteur Y n'a pas de nom.
Le numéro de série du volume est B2B7-2241

Répertoire de Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\pipenv

```
22/01/2018 10:39    <DIR>          .
22/01/2018 10:39    <DIR>          ..
22/01/2018 08:23            250 app.py
22/01/2018 10:11            438 Dockerfile
22/01/2018 10:39            8 130 pipenv.rst
22/01/2018 08:23            129 Pipfile
22/01/2018 08:23            2 580 Pipfile.lock
22/01/2018 08:23            415 Readme.md
6 fichier(s)           11 942 octets
2 Rép(s)   20 168 241 152 octets libres
```

24.13.2 Réécriture du fichier Dockerfile

See also:

- <https://github.com/pypa/pipenv/blob/master/Dockerfile>

On part de la recommandation officielle de Kenneth Reitz.

```

1 # https://github.com/pypa/pipenv/blob/master/Dockerfile
2 FROM python:3.6
3
4 # -- Install Pipenv:
5 RUN set -ex && pip install pipenv --upgrade
6
7 # -- Install Application into container:
8 RUN set -ex && mkdir /app
9
10 WORKDIR /app
11
12 # -- Adding Pipfiles
13 COPY Pipfile Pipfile
14 # COPY Pipfile.lock Pipfile.lock
15
16 # -- Install dependencies:
17 RUN set -ex && pipenv install --deploy --system
18
19 COPY app.py /app
20
21 CMD ["python", "app.py"]
22

```

24.13.3 app.py

```

1 """ This is a very basic flask server"""
2 from flask import Flask
3
4 app = Flask(__name__)
5
6 @app.route("/")
7 def hello():
8     """docstring"""
9     return "Hello World!"
10
11
12 if __name__ == '__main__':
13     app.run(host="0.0.0.0", debug = True)

```

24.13.4 docker build -t docker-pipenv-sample . : construction de l'image

```
C:/projects_id3/docker_projects/docker-pipenv-sample>docker build -t docker-pipenv-
sample .
```

```
Sending build context to Docker daemon 78.34kB
Step 1/8 : FROM python:3.6
```

(continues on next page)

(continued from previous page)

```

3.6: Pulling from library/python
Digest: sha256:98149ed5f37f48ea3fad26ae6c0042dd2b08228d58edc95ef0fce35f1b3d9e9f
Status: Downloaded newer image for python:3.6
--> cle459c00dc3
Step 2/8 : RUN set -ex && pip install pipenv --upgrade
--> Running in 21e4931d7ee4
+ pip install pipenv --upgrade
Collecting pipenv
  Downloading pipenv-9.0.3.tar.gz (3.9MB)
Collecting virtualenv (from pipenv)
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
Collecting pew>=0.1.26 (from pipenv)
  Downloading pew-1.1.2-py2.py3-none-any.whl
Requirement already up-to-date: pip>=9.0.1 in /usr/local/lib/python3.6/site-packages_
→ (from pipenv)
Collecting requests>2.18.0 (from pipenv)
  Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
Collecting flake8>=3.0.0 (from pipenv)
  Downloading flake8-3.5.0-py2.py3-none-any.whl (69kB)
Collecting urllib3>=1.21.1 (from pipenv)
  Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
Collecting virtualenv-clone>=0.2.5 (from pew>=0.1.26->pipenv)
  Downloading virtualenv-clone-0.2.6.tar.gz
Collecting setuptools>=17.1 (from pew>=0.1.26->pipenv)
  Downloading setuptools-38.4.0-py2.py3-none-any.whl (489kB)
Collecting certifi>=2017.4.17 (from requests>2.18.0->pipenv)
  Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
Collecting chardet<3.1.0,>=3.0.2 (from requests>2.18.0->pipenv)
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
Collecting idna<2.7,>=2.5 (from requests>2.18.0->pipenv)
  Downloading idna-2.6-py2.py3-none-any.whl (56kB)
Collecting mccabe<0.7.0,>=0.6.0 (from flake8>=3.0.0->pipenv)
  Downloading mccabe-0.6.1-py2.py3-none-any.whl
Collecting pycodestyle<2.4.0,>=2.0.0 (from flake8>=3.0.0->pipenv)
  Downloading pycodestyle-2.3.1-py2.py3-none-any.whl (45kB)
Collecting pyflakes<1.7.0,>=1.5.0 (from flake8>=3.0.0->pipenv)
  Downloading pyflakes-1.6.0-py2.py3-none-any.whl (227kB)
Building wheels for collected packages: pipenv, virtualenv-clone
  Running setup.py bdist_wheel for pipenv: started
  Running setup.py bdist_wheel for pipenv: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/78/cf/b7/
→549d89ddbf1cf3da825b97b730a7e1ac75602de9865d036e
  Running setup.py bdist_wheel for virtualenv-clone: started
  Running setup.py bdist_wheel for virtualenv-clone: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/24/51/ef/
→93120d304d240b4b6c2066454250a1626e04f73d34417b956d
Successfully built pipenv virtualenv-clone
Installing collected packages: virtualenv, virtualenv-clone, setuptools, pew, urllib3,
→ certifi, chardet, idna, requests, mccabe, pycodestyle, pyflakes, flake8, pipenv
  Found existing installation: setuptools 38.2.4
    Uninstalling setuptools-38.2.4:
      Successfully uninstalled setuptools-38.2.4
Successfully installed certifi-2018.1.18 chardet-3.0.4 flake8-3.5.0 idna-2.6 mccabe-0.
→6.1 pew-1.1.2 pipenv-9.0.3 pycodestyle-2.3.1 pyflakes-1.6.0 requests-2.18.4_
→setuptools-38.4.0 urllib3-1.22 virtualenv-15.1.0 virtualenv-clone-0.2.6
Removing intermediate container 21e4931d7ee4
--> 0b1272e6e1c6

```

(continues on next page)

(continued from previous page)

```

Step 3/8 : RUN set -ex && mkdir /app
--> Running in 21153ac29a7f
+ mkdir /app
Removing intermediate container 21153ac29a7f
--> 1f95b3a89e78
Step 4/8 : WORKDIR /app
Removing intermediate container d235da053693
--> c40c0a57be56
Step 5/8 : COPY Pipfile Pipfile
--> 72c20255a55d
Step 6/8 : COPY Pipfile.lock Pipfile.lock
--> 7f022488626e
Step 7/8 : RUN set -ex && pipenv install --deploy --system
--> Running in 7535ac2a9610
+ pipenv install --deploy --system
Installing dependencies from Pipfile.lock (d3d473)...
Removing intermediate container 7535ac2a9610
--> 7366de78a2f1
Step 8/8 : COPY . /app
--> 5c977e084023
Successfully built 5c977e084023
Successfully tagged docker-pipenv-sample:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows
--> Docker host.
All files and directories added to build context will have '-rwxr-xr-x' permissions.
It is recommended to double check and reset permissions for sensitive files and
--> directories.

```

24.13.5 docker run -p 5000:5000 docker-pipenv-sample

```
C:/projects_id3/docker_projects/docker-pipenv-sample>docker run -p 5000:5000 docker-
--> pipenv-sample
```

```
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 153-767-505
```

24.13.6 http://localhost:5000/

24.13.7 docker ps

```
Y:/projects_id3/P5N001/XLOGCA135_tutorial_docker/tutorial_docker/tutoriels/pipenv>
--> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	
STATUS	PORTS	NAMES		
b9bf3fb859	docker-pipenv-sample	"python app.py"	4 minutes ago	Up
--> 4 minutes	0.0.0.0:5000->5000/tcp	condescending_hypatia		

24.13.8 docker exec -it 1a0a3dc7924d bash

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\pipenv>
→docker exec -it b9bf3fbbb859 bash
```

```
root@b9bf3fbbb859:/app# ls -als
```

```
4 drwxr-xr-x 1 root root 4096 Jan 22 09:44 .
4 drwxr-xr-x 1 root root 4096 Jan 22 09:45 ..
4 -rwxr-xr-x 1 root root 129 Jan 22 07:23 Pipfile
4 -rwxr-xr-x 1 root root 2580 Jan 22 07:23 Pipfile.lock
4 -rwxr-xr-x 1 root root 248 Jan 22 09:43 app.py
```

```
root@1a0a3dc7924d:/app# ps -ef | grep python
```

root	1	0	0 08:42 ?	00:00:00	python app.py
root	7	1	0 08:42 ?	00:00:10	/usr/local/bin/python app.py

24.13.9 docker rm 1a0a3dc7924d: suppression du conteneur à l'arrêt

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\pipenv>
→docker rm 1a0a3dc7924d
```

```
1a0a3dc7924d
```

24.13.10 docker rmi docker-pipenv-sample: suppression de l'image

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\pipenv>
→docker rmi docker-pipenv-sample
```

```
Untagged: docker-pipenv-sample:latest
Deleted: sha256:f7cb7fa32f377aa356791f7149f8f21b2b668e6ce5011dc338cb8ea7c58778b9
Deleted: sha256:91953983b1e474e3aff636101c4625d825c8a54044a7a44026d8a4a049efa5d7
Deleted: sha256:b08673d3c06b5d6c576e64d0c87f1d09d53355ae8f416d9e12b125bb78425721
```

24.14 play with docker

See also:

- <https://training.play-with-docker.com/>

Contents

- *play with docker*
 - *Docker for IT Pros and System Administrators Stage 1*
 - *Docker for Beginners - Linux*

24.14.1 Docker for IT Pros and System Administrators Stage 1

See also:

- <https://training.play-with-docker.com/ops-stage1/>

24.14.2 Docker for Beginners - Linux

See also:

- <https://training.play-with-docker.com/beginner-linux/>

24.15 Centos7

See also:

- <http://www.codeghar.com/blog/install-latest-python-on-centos-7.html>

Contents

- *Centos7*
 - *Plan de travail*
 - *yum update*
 - *yum install -y https://centos7.iuscommunity.org/ius-release.rpm*
 - *yum install -y python36u python36u-libs python36u-devel python36u-pip*
 - *python3.6*
 - *yum install which*
 - *which pip3.6*
 - *docker build -t id3centos7:1 .*
 - *docker images*
 - *docker run --name test -it id3centos7:1*
 - *Probleme avec regex*
 - *yum install gcc*
 - *yum install openldap-devel*
 - *pip install pyldap*
 - *Nouveau fichier Dockerfile*
 - * *Dockerfile*
 - * *which python3.6*
 - * *python3.6 -m pip install pipenv*
 - *Nouveau Dockerfile*
 - * *Dockerfile*

```
* docker build -t id3centos7:0.1.1 .
- Nouveau fichier Dockerfile
  * Dockerfile
  * Construction de l'image docker build -t id3centos7:0.1.2 .
  * docker run -name id3centos7.1.2 -it id3centos7:0.1.2
- Nouveau dockerfile
  * Dockerfile
- Nouveau fichier Dockerfile
  * Dockerfile
- Nouveau fichier Dockerfile
```

24.15.1 Plan de travail

- récupérer une image centos:7
- yum update
- yum install -y <https://centos7.iuscommunity.org/ius-release.rpm>
- yum install -y python36u python36u-lib python36u-devel python36u-pip
- yum install which
- yum install openldap-devel
- pip3.6 install pipenv

24.15.2 yum update

```
[root@20c8bd8c86f4 intranet] # yum update
```

```
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: ftp.pasteur.fr
* epel: pkg.adfinis-sygroup.ch
* extras: mirror.plusserver.com
* ius: mirror.slu.cz
* updates: ftp.ciril.fr
Resolving Dependencies
--> Running transaction check
--> Package bind-license.noarch 32:9.9.4-51.el7_4.1 will be updated
--> Package bind-license.noarch 32:9.9.4-51.el7_4.2 will be an update
--> Package binutils.x86_64 0:2.25.1-32.base.el7_4.1 will be updated
--> Package binutils.x86_64 0:2.25.1-32.base.el7_4.2 will be an update
--> Package epel-release.noarch 0:7-9 will be updated
--> Package epel-release.noarch 0:7-11 will be an update
--> Package kmod.x86_64 0:20-15.el7_4.6 will be updated
--> Package kmod.x86_64 0:20-15.el7_4.7 will be an update
--> Package kmod-libs.x86_64 0:20-15.el7_4.6 will be updated
--> Package kmod-libs.x86_64 0:20-15.el7_4.7 will be an update
```

(continues on next page)

(continued from previous page)

```

--> Package kpartx.x86_64 0:0.4.9-111.el7 will be updated
--> Package kpartx.x86_64 0:0.4.9-111.el7_4.2 will be an update
--> Package libdbd.x86_64 0:5.3.21-20.el7 will be updated
--> Package libdbd.x86_64 0:5.3.21-21.el7_4 will be an update
--> Package libdbd-utils.x86_64 0:5.3.21-20.el7 will be updated
--> Package libdbd-utils.x86_64 0:5.3.21-21.el7_4 will be an update
--> Package systemd.x86_64 0:219-42.el7_4.4 will be updated
--> Package systemd.x86_64 0:219-42.el7_4.7 will be an update
--> Package systemd-libs.x86_64 0:219-42.el7_4.4 will be updated
--> Package systemd-libs.x86_64 0:219-42.el7_4.7 will be an update
--> Package tzdata.noarch 0:2017c-1.el7 will be updated
--> Package tzdata.noarch 0:2018c-1.el7 will be an update
--> Package yum.noarch 0:3.4.3-154.el7.centos will be updated
--> Package yum.noarch 0:3.4.3-154.el7.centos.1 will be an update
--> Finished Dependency Resolution

```

Dependencies Resolved

Package	Arch	Version
	Repository	Size
Updating:		
bind-license	noarch	32:9.9.4-51.el7_
↳ 4.2	updates	84 k
binutils	x86_64	2.25.1-32.base.
↳ el7_4.2	updates	5.4 M
epel-release	noarch	7-11
↳	epel	15 k
kmmod	x86_64	20-15.el7_4.7
↳	updates	121 k
kmmod-libs	x86_64	20-15.el7_4.7
↳	updates	50 k
kpartx	x86_64	0.4.9-111.el7_4.
↳ 2	updates	73 k
libdbd	x86_64	5.3.21-21.el7_4
↳	updates	719 k
libdbd-utils	x86_64	5.3.21-21.el7_4
↳	updates	132 k
systemd	x86_64	219-42.el7_4.7
↳	updates	5.2 M
systemd-libs	x86_64	219-42.el7_4.7
↳	updates	376 k
tzdata	noarch	2018c-1.el7
↳	updates	479 k
yum	noarch	3.4.3-154.el7.
↳ centos.1	updates	1.2 M

Transaction Summary

Upgrade 12 Packages

Total download size: 14 M

Is this ok [y/d/N]: y

Downloading packages:

Delta RPMs disabled because /usr/bin/applydeltarpm not installed.

(1/12): bind-license-9.9.4-51.el7_4.2.noarch.rpm

→ | 84 kB 00:00:00 (continues on next page)

(continued from previous page)

```
(2/12) : kmod-libs-20-15.el7_4.7.x86_64.rpm | 50 kB 00:00:00
(3/12) : kmod-20-15.el7_4.7.x86_64.rpm | 121 kB 00:00:00
warning: /var/cache/yum/x86_64/7/epel/packages/epel-release-7-11.noarch.rpm: Header V3 RSA/SHA256 Signature, key ID 352c64e5: NOKEY
Public key for epel-release-7-11.noarch.rpm is not installed
(4/12) : epel-release-7-11.noarch.rpm | 15 kB 00:00:00
(5/12) : libdb-utils-5.3.21-21.el7_4.x86_64.rpm | 132 kB 00:00:00
(6/12) : kpartx-0.4.9-111.el7_4.2.x86_64.rpm | 73 kB 00:00:00
(7/12) : libdb-5.3.21-21.el7_4.x86_64.rpm | 719 kB 00:00:01
(8/12) : tzdata-2018c-1.el7.noarch.rpm | 479 kB 00:00:01
(9/12) : systemd-libs-219-42.el7_4.7.x86_64.rpm | 376 kB 00:00:02
(10/12) : yum-3.4.3-154.el7.centos.1.noarch.rpm | 1.2 MB 00:00:03
(11/12) : binutils-2.25.1-32.base.el7_4.2.x86_64.rpm | 5.4 MB 00:00:10
(12/12) : systemd-219-42.el7_4.7.x86_64.rpm | 5.2 MB 00:00:10
-----
Total 1.2 MB/s | 14 MB 00:00:11
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Importing GPG key 0x352C64E5:
Userid      : "Fedora EPEL (7) <epel@fedoraproject.org>"
Fingerprint: 91e9 7d7c 4a5e 96f1 7f3e 888f 6a2f aea2 352c 64e5
Package     : epel-release-7-9.noarch (@extras)
From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Is this ok [y/N]: y
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Updating   : libdb-5.3.21-21.el7_4.x86_64          1/24
Updating   : binutils-2.25.1-32.base.el7_4.2.x86_64    2/24
Updating   : kmod-20-15.el7_4.7.x86_64            3/24
Updating   : systemd-libs-219-42.el7_4.7.x86_64      4/24
Updating   : kmod-libs-20-15.el7_4.7.x86_64        5/24
Updating   : systemd-219-42.el7_4.7.x86_64          6/24
Updating   : libdb-utils-5.3.21-21.el7_4.x86_64     7/24
Updating   : yum-3.4.3-154.el7.centos.1.noarch      8/24
Updating   : 32:bind-license-9.9.4-51.el7_4.2.noarch 9/24

```

9/24 (continues on next page)

(continued from previous page)

Updating	:	tzdata-2018c-1.el7.noarch		
↳			10/24	█
Updating	:	kpartx-0.4.9-111.el7_4.2.x86_64		█
↳			11/24	█
Updating	:	epel-release-7-11.noarch		█
↳			12/24	█
Cleanup	:	systemd-219-42.el7_4.4.x86_64		█
↳			13/24	█
Cleanup	:	kmod-20-15.el7_4.6.x86_64		█
↳			14/24	█
Cleanup	:	libdb-utils-5.3.21-20.el7.x86_64		█
↳			15/24	█
Cleanup	:	yum-3.4.3-154.el7.centos.noarch		█
↳			16/24	█
Cleanup	:	32:bind-license-9.9.4-51.el7_4.1.noarch		█
↳			17/24	█
Cleanup	:	tzdata-2017c-1.el7.noarch		█
↳			18/24	█
Cleanup	:	epel-release-7-9.noarch		█
↳			19/24	█
Cleanup	:	libdb-5.3.21-20.el7.x86_64		█
↳			20/24	█
Cleanup	:	binutils-2.25.1-32.base.el7_4.1.x86_64		█
↳			21/24	█
Cleanup	:	kmod-libs-20-15.el7_4.6.x86_64		█
↳			22/24	█
Cleanup	:	systemd-libs-219-42.el7_4.4.x86_64		█
↳			23/24	█
Cleanup	:	kpartx-0.4.9-111.el7.x86_64		█
↳			24/24	█
Verifying	:	kmod-20-15.el7_4.7.x86_64		█
↳			1/24	█
Verifying	:	kmod-libs-20-15.el7_4.7.x86_64		█
↳			2/24	█
Verifying	:	libdb-utils-5.3.21-21.el7_4.x86_64		█
↳			3/24	█
Verifying	:	systemd-219-42.el7_4.7.x86_64		█
↳			4/24	█
Verifying	:	epel-release-7-11.noarch		█
↳			5/24	█
Verifying	:	kpartx-0.4.9-111.el7_4.2.x86_64		█
↳			6/24	█
Verifying	:	tzdata-2018c-1.el7.noarch		█
↳			7/24	█
Verifying	:	32:bind-license-9.9.4-51.el7_4.2.noarch		█
↳			8/24	█
Verifying	:	systemd-libs-219-42.el7_4.7.x86_64		█
↳			9/24	█
Verifying	:	binutils-2.25.1-32.base.el7_4.2.x86_64		█
↳			10/24	█
Verifying	:	libdb-5.3.21-21.el7_4.x86_64		█
↳			11/24	█
Verifying	:	yum-3.4.3-154.el7.centos.1.noarch		█
↳			12/24	█
Verifying	:	epel-release-7-9.noarch		█
↳			13/24	█
Verifying	:	binutils-2.25.1-32.base.el7_4.1.x86_64		█
↳			14/24	(continues on next page) █

(continued from previous page)

```

Verifying : 32:bind-license-9.9.4-51.el7_4.1.noarch                                ↵
↳                                                 15/24
Verifying : systemd-libs-219-42.el7_4.4.x86_64                                ↵
↳                                                 16/24
Verifying : kmod-20-15.el7_4.6.x86_64                                         ↵
↳                                                 17/24
Verifying : systemd-219-42.el7_4.4.x86_64                                         ↵
↳                                                 18/24
Verifying : libdb-utils-5.3.21-20.el7.x86_64                                    ↵
↳                                                 19/24
Verifying : kmod-libs-20-15.el7_4.6.x86_64                                     ↵
↳                                                 20/24
Verifying : tzdata-2017c-1.el7.noarch                                         ↵
↳                                                 21/24
Verifying : kpartx-0.4.9-111.el7.x86_64                                       ↵
↳                                                 22/24
Verifying : yum-3.4.3-154.el7.centos.noarch                                    ↵
↳                                                 23/24
Verifying : libdb-5.3.21-20.el7.x86_64                                         ↵
↳                                                 24/24

Updated:
bind-license.noarch 32:9.9.4-51.el7_4.2      binutils.x86_64 0:2.25.1-32.base.el7_4.2 ↵
↳epel-release.noarch 0:7-11                  kmod.x86_64 0:20-15.el7_4.7
kmod-libs.x86_64 0:20-15.el7_4.7           kpartx.x86_64 0:0.4.9-111.el7_4.2
↳libdb.x86_64 0:5.3.21-21.el7_4          libdb-utils.x86_64 0:5.3.21-21.el7_4
systemd.x86_64 0:219-42.el7_4.7           systemd-libs.x86_64 0:219-42.el7_4.7
↳tzdata.noarch 0:2018c-1.el7               yum.noarch 0:3.4.3-154.el7.centos.1

Complete!
[root@20c8bd8c86f4 intranet]#

```

24.15.3 yum install -y https://centos7.iuscommunity.org/ius-release.rpm

```
[root@20c8bd8c86f4 /]# yum install -y https://centos7.iuscommunity.org/ius-release.rpm
```

Loaded plugins: fastestmirror, ovl	
ius-release.rpm	8.1 kB 00:00:00
↳	
Examining /var/tmp/yum-root-KswZN7/ius-release.rpm: ius-release-1.0-15.ius.centos7.	
↳noarch	
Marking /var/tmp/yum-root-KswZN7/ius-release.rpm to be installed	
Resolving Dependencies	
--> Running transaction check	
--> Package ius-release.noarch 0:1.0-15.ius.centos7 will be installed	
--> Processing Dependency: epel-release = 7 for package: ius-release-1.0-15.ius.	
↳centos7.noarch	
base	3.6 kB 00:00:00
↳	
extras	3.4 kB 00:00:00
↳	
updates	3.4 kB 00:00:00
↳	
(1/4): extras/7/x86_64/primary_db	166 kB 00:00:00
↳	

(continues on next page)

(continued from previous page)

```
(2/4) : base/7/x86_64/group_gz | 156 kB 00:00:01
(3/4) : updates/7/x86_64/primary_db | 6.0 MB 00:00:04
(4/4) : base/7/x86_64/primary_db | 5.7 MB 00:00:14
Determining fastest mirrors
* base: ftp.pasteur.fr
* extras: mirror.plusserver.com
* updates: ftp.ciril.fr
--> Running transaction check
--> Package epel-release.noarch 0:7-9 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version          Size
Repository
=====
Installing:
ius-release      noarch   1.0-15.ius.    8.5 k
centos7          /ius-release
Installing for dependencies:
epel-release     noarch   7-9             14 k
extras

Transaction Summary
=====
Install 1 Package (+1 Dependent package)

Total size: 23 k
Total download size: 14 k
Installed size: 33 k
Downloading packages:
warning: /var/cache/yum/x86_64/7/extras/packages/epel-release-7-9.noarch.rpm: Header
  ↵V3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEYB/s | 0 B  ---- ETA
Public key for epel-release-7-9.noarch.rpm is not installed
epel-release-7-9.noarch.rpm
  ↵                                         | 14 kB 00:00:00
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Importing GPG key 0xF4A80EB5:
Userid      : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
Package     : centos-release-7-4.1708.el7.centos.x86_64 (@CentOS)
From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : epel-release-7-9.noarch
  ↵                                         1/2
Installing : ius-release-1.0-15.ius.centos7.noarch
  ↵                                         2/2
Verifying   : ius-release-1.0-15.ius.centos7.noarch
  ↵                                         1/2
Verifying   : epel-release-7-9.noarch
  ↵                                         2/2
```

2/2 (continues on next page)

(continued from previous page)

```
Installed:  
ius-release.noarch 0:1.0-15.ius.centos7  
  
Dependency Installed:  
epel-release.noarch 0:7-9  
  
Complete!
```

24.15.4 yum install -y python36u python36u-libs python36u-devel python36u-pip

```
[root@20c8bd8c86f4 /]# yum install -y python36u python36u-libs python36u-devel  
→python36u-pip
```

```
Loaded plugins: fastestmirror, ovl  
epel/x86_64/metalink  
→  
epel  
→  
ius  
→  
(1/4): epel/x86_64/group_gz  
→  
(2/4): ius/x86_64/primary_db  
→  
(3/4): epel/x86_64/primary_db  
→  
(4/4): epel/x86_64/updateinfo  
→  
Loading mirror speeds from cached hostfile  
* base: ftp.pasteur.fr  
* epel: ftp-stud.hs-esslingen.de  
* extras: mirror.plusserver.com  
* ius: mirror.team-cymru.org  
* updates: ftp.ciril.fr  
Resolving Dependencies  
--> Running transaction check  
---> Package python36u.x86_64 0:3.6.4-1.ius.centos7 will be installed  
---> Package python36u-devel.x86_64 0:3.6.4-1.ius.centos7 will be installed  
---> Package python36u-libs.x86_64 0:3.6.4-1.ius.centos7 will be installed  
---> Package python36u-pip.noarch 0:9.0.1-1.ius.centos7 will be installed  
---> Processing Dependency: python36u-setuptools for package: python36u-pip-9.0.1-1.  
→ius.centos7.noarch  
--> Running transaction check  
---> Package python36u-setuptools.noarch 0:36.6.0-1.ius.centos7 will be installed  
--> Finished Dependency Resolution  
  
Dependencies Resolved  
  
=====
```

Package →	Arch Repository	Version Size
Installing:		

(continues on next page)

(continued from previous page)

```

python36u                               x86_64          3.6.4-1.
  ↵ius.centos7                         ius              56 k
python36u-devel                          x86_64          3.6.4-1.
  ↵ius.centos7                         ius             839 k
python36u-libs                           x86_64          3.6.4-1.
  ↵ius.centos7                         ius              8.7 M
python36u-pip                            noarch         9.0.1-1.
  ↵ius.centos7                         ius             1.8 M
Installing for dependencies:
python36u-setuptools                     noarch         36.6.0-1.
  ↵ius.centos7                         ius             587 k

Transaction Summary
=====
Install 4 Packages (+1 Dependent package)

Total download size: 12 M
Installed size: 53 M
Downloading packages:
warning: /var/cache/yum/x86_64/7/ius/packages/python36u-3.6.4-1.ius.centos7.x86_64.
  ↵rpm: Header V4 DSA/SHA1 Signature, key ID 9cd4953f: NOKEY2 kB --::-- ETA
Public key for python36u-3.6.4-1.ius.centos7.x86_64.rpm is not installed
(1/5): python36u-3.6.4-1.ius.centos7.x86_64.rpm
  ↵                                         | 56 kB  00:00:00
(2/5): python36u-setuptools-36.6.0-1.ius.centos7.noarch.rpm
  ↵                                         | 587 kB  00:00:03
(3/5): python36u-pip-9.0.1-1.ius.centos7.noarch.rpm
  ↵                                         | 1.8 MB  00:00:03
(4/5): python36u-devel-3.6.4-1.ius.centos7.x86_64.rpm
  ↵                                         | 839 kB  00:00:06
(5/5): python36u-libs-3.6.4-1.ius.centos7.x86_64.rpm
  ↵                                         | 8.7 MB  00:00:28
-----
  ↵
Total                                         432 kB/s | 12 MB  00:00:28
  ↵
Retrieving key from file:///etc/pki/rpm-gpg/IUS-COMMUNITY-GPG-KEY
Importing GPG key 0x9CD4953F:
Userid      : "IUS Community Project <coredev@iuscommunity.org>"
Fingerprint: 8b84 6e3a b3fe 6462 74e8 670f da22 1cdf 9cd4 953f
Package     : ius-release-1.0-15.ius.centos7.noarch (installed)
From       : /etc/pki/rpm-gpg/IUS-COMMUNITY-GPG-KEY
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : python36u-libs-3.6.4-1.ius.centos7.x86_64
  ↵                                                 1/5
Installing : python36u-3.6.4-1.ius.centos7.x86_64
  ↵                                                 2/5
Installing : python36u-setuptools-36.6.0-1.ius.centos7.noarch
  ↵                                                 3/5
Installing : python36u-pip-9.0.1-1.ius.centos7.noarch
  ↵                                                 4/5
Installing : python36u-devel-3.6.4-1.ius.centos7.x86_64
  ↵                                                 5/5
Verifying   : python36u-setuptools-36.6.0-1.ius.centos7.noarch
  ↵

```

1/5 (continues on next page)

(continued from previous page)

```

Verifying : python36u-pip-9.0.1-1.ius.centos7.noarch          ↵
↳                                                 2/5
Verifying : python36u-3.6.4-1.ius.centos7.x86_64            ↵
↳                                                 3/5
Verifying : python36u-libs-3.6.4-1.ius.centos7.x86_64        ↵
↳                                                 4/5
Verifying : python36u-devel-3.6.4-1.ius.centos7.x86_64       ↵
↳                                                 5/5

Installed:
python36u.x86_64 0:3.6.4-1.ius.centos7           python36u-devel.x86_64 0:3.6.4-1.
↳ ius.centos7           python36u-libs.x86_64 0:3.6.4-1.ius.centos7
python36u-pip.noarch 0:9.0.1-1.ius.centos7

Dependency Installed:
python36u-setuptools.noarch 0:36.6.0-1.ius.centos7

Complete!
[root@20c8bd8c86f4 /]#

```

24.15.5 python3.6

```
[root@20c8bd8c86f4 /]# python3.6
Python 3.6.4 (default, Dec 19 2017, 14:48:12)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

24.15.6 yum install which

```
[root@20c8bd8c86f4 /]# yum install which
```

```

Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: ftp.pasteur.fr
* epel: repo.boun.edu.tr
* extras: mirror.plusserver.com
* ius: mirror.its.dal.ca
* updates: ftp.ciril.fr
Resolving Dependencies
--> Running transaction check
--> Package which.x86_64 0:2.20-7.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
```

```
=====
Package           Arch      Repository      Version      Size
↳
           Repository
=====
Installing:
which           x86_64      base          2.20-7.el7   ↵
↳
           base          41 k
(continues on next page)
```

(continued from previous page)

```

Transaction Summary
=====
Install 1 Package

Total download size: 41 k
Installed size: 75 k
Is this ok [y/d/N]: y
Downloading packages:
which-2.20-7.el7.x86_64.rpm
| 41 kB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : which-2.20-7.el7.x86_64
| 1/1
install-info: No such file or directory for /usr/share/info/which.info.gz
  Verifying : which-2.20-7.el7.x86_64
| 1/1

Installed:
  which.x86_64 0:2.20-7.el7

Complete!
[root@20c8bd8c86f4 ~]# which python3.6
/usr/bin/python3.6

[root@20c8bd8c86f4 ~]# which python3.6
/usr/bin/python3.6

```

24.15.7 which pip3.6

```

[root@20c8bd8c86f4 ~]# which pip3.6
/usr/bin/pip3.6

[root@20c8bd8c86f4 ~]# pip3.6 install pipenv

Collecting pipenv
  Downloading pipenv-9.0.3.tar.gz (3.9MB)
  100% | #####| 3.9MB 291kB/s
Collecting virtualenv (from pipenv)
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
  100% | #####| 1.8MB 610kB/s
Collecting pew>=0.1.26 (from pipenv)
  Downloading pew-1.1.2-py2.py3-none-any.whl
Requirement already satisfied: pip>=9.0.1 in /usr/lib/python3.6/site-packages (from
pipenv)
Collecting requests>2.18.0 (from pipenv)
  Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
  100% | #####| 92kB 1.1MB/s
Collecting flake8>=3.0.0 (from pipenv)

```

(continues on next page)

(continued from previous page)

```
Downloading flake8-3.5.0-py2.py3-none-any.whl (69kB)
100% | #####| 71kB 2.8MB/s
Collecting urllib3>=1.21.1 (from pipenv)
Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
100% | #####| 133kB 2.0MB/s
Requirement already satisfied: setuptools>=17.1 in /usr/lib/python3.6/site-packages
→ (from pew>=0.1.26->pipenv)
Collecting virtualenv-clone>=0.2.5 (from pew>=0.1.26->pipenv)
Downloading virtualenv-clone-0.2.6.tar.gz
Collecting certifi>=2017.4.17 (from requests>2.18.0->pipenv)
Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
100% | #####| 153kB 1.0MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests>2.18.0->pipenv)
Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
100% | #####| 143kB 2.4MB/s
Collecting idna<2.7,>=2.5 (from requests>2.18.0->pipenv)
Downloading idna-2.6-py2.py3-none-any.whl (56kB)
100% | #####| 61kB 920kB/s
Collecting mccabe<0.7.0,>=0.6.0 (from flake8>=3.0.0->pipenv)
Downloading mccabe-0.6.1-py2.py3-none-any.whl
Collecting pycodestyle<2.4.0,>=2.0.0 (from flake8>=3.0.0->pipenv)
Downloading pycodestyle-2.3.1-py2.py3-none-any.whl (45kB)
100% | #####| 51kB 2.2MB/s
Collecting pyflakes<1.7.0,>=1.5.0 (from flake8>=3.0.0->pipenv)
Downloading pyflakes-1.6.0-py2.py3-none-any.whl (227kB)
100% | #####| 235kB 2.3MB/s
Installing collected packages: virtualenv, virtualenv-clone, pew, certifi, urllib3,
→ chardet, idna, requests, mccabe, pycodestyle, pyflakes, flake8, pipenv
Running setup.py install for virtualenv-clone ... done
Running setup.py install for pipenv ... done
Successfully installed certifi-2018.1.18 chardet-3.0.4 flake8-3.5.0 idna-2.6 mccabe-0.
→ 6.1 pew-1.1.2 pipenv-9.0.3 pycodestyle-2.3.1 pyflakes-1.6.0 requests-2.18.4 urllib3-
→ 1.22 virtualenv-15.1.0 virtualenv-clone-0.2.6
```

```
(activate) [root@20c8bd8c86f4 intranet]# pip install django
Collecting django
  Downloading Django-2.0.2-py3-none-any.whl (7.1MB)
    100% | #####| 7.1MB 205kB/s
Collecting pytz (from django)
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% | #####| 512kB 1.5MB/s
Installing collected packages: pytz, django
Successfully installed django-2.0.2 pytz-2017.3
```

24.15.8 docker build -t id3centos7:1 .

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:1 .
```

```
Sending build context to Docker daemon 37.38kB
Step 1/5 : FROM centos:7
--> ff426288ea90
Step 2/5 : RUN yum update -y
--> Running in bd9bc627aeeb
```

(continues on next page)

(continued from previous page)

```

Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
* base: centos.quelquesmots.fr
* extras: fr.mirror.babylon.network
* updates: fr.mirror.babylon.network
Resolving Dependencies
--> Running transaction check
--> Package bind-license.noarch 32:9.9.4-51.el7_4.1 will be updated
--> Package bind-license.noarch 32:9.9.4-51.el7_4.2 will be an update
--> Package binutils.x86_64 0:2.25.1-32.base.el7_4.1 will be updated
--> Package binutils.x86_64 0:2.25.1-32.base.el7_4.2 will be an update
--> Package kmod.x86_64 0:20-15.el7_4.6 will be updated
--> Package kmod.x86_64 0:20-15.el7_4.7 will be an update
--> Package kmod-libs.x86_64 0:20-15.el7_4.6 will be updated
--> Package kmod-libs.x86_64 0:20-15.el7_4.7 will be an update
--> Package kpartx.x86_64 0:0.4.9-111.el7 will be updated
--> Package kpartx.x86_64 0:0.4.9-111.el7_4.2 will be an update
--> Package libdb.x86_64 0:5.3.21-20.el7 will be updated
--> Package libdb.x86_64 0:5.3.21-21.el7_4 will be an update
--> Package libdb-utils.x86_64 0:5.3.21-20.el7 will be updated
--> Package libdb-utils.x86_64 0:5.3.21-21.el7_4 will be an update
--> Package systemd.x86_64 0:219-42.el7_4.4 will be updated
--> Package systemd.x86_64 0:219-42.el7_4.7 will be an update
--> Package systemd-libs.x86_64 0:219-42.el7_4.4 will be updated
--> Package systemd-libs.x86_64 0:219-42.el7_4.7 will be an update
--> Package tzdata.noarch 0:2017c-1.el7 will be updated
--> Package tzdata.noarch 0:2018c-1.el7 will be an update
--> Package yum.noarch 0:3.4.3-154.el7.centos will be updated
--> Package yum.noarch 0:3.4.3-154.el7.centos.1 will be an update
--> Finished Dependency Resolution

```

Dependencies Resolved

```
=====
Package           Arch      Version            Repository      Size
=====
Updating:
bind-license     noarch   32:9.9.4-51.el7_4.2    updates        84 k
binutils          x86_64   2.25.1-32.base.el7_4.2  updates       5.4 M
kmod              x86_64   20-15.el7_4.7          updates      121 k
kmod-libs         x86_64   20-15.el7_4.7          updates      50 k
kpartx            x86_64   0.4.9-111.el7_4.2    updates      73 k
libdb              x86_64   5.3.21-21.el7_4        updates     719 k
libdb-utils        x86_64   5.3.21-21.el7_4        updates     132 k
systemd            x86_64   219-42.el7_4.7        updates      5.2 M
systemd-libs       x86_64   219-42.el7_4.7        updates     376 k
tzdata             noarch   2018c-1.el7          updates     479 k
yum                noarch   3.4.3-154.el7.centos.1  updates     1.2 M
```

Transaction Summary

```
=====
Upgrade 11 Packages
```

Total download size: 14 M

Downloading packages:

Delta RPMs disabled because /usr/bin/applydeltarpm not installed.

```
warning: /var/cache/yum/x86_64/7/updates/packages/kmod-libs-20-15.el7_4.7.x86_64.rpm: [REDACTED]
Header V3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEY
```

(continues on next page)

(continued from previous page)

Public key for kmod-libs-20-15.el7_4.7.x86_64.rpm is not installed	

Total	1.6 MB/s 14 MB 00:08
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7	
Importing GPG key 0xF4A80EB5:	
Userid : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"	
Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5	
Package : centos-release-7-4.1708.el7.centos.x86_64 (@CentOS)	
From : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7	
Running transaction check	
Running transaction test	
Transaction test succeeded	
Running transaction	
Updating : libdb-5.3.21-21.el7_4.x86_64	1/22
Updating : binutils-2.25.1-32.base.el7_4.2.x86_64	2/22
Updating : kmod-20-15.el7_4.7.x86_64	3/22
Updating : systemd-libs-219-42.el7_4.7.x86_64	4/22
Updating : kmod-libs-20-15.el7_4.7.x86_64	5/22
Updating : systemd-219-42.el7_4.7.x86_64	6/22
Updating : libdb-utils-5.3.21-21.el7_4.x86_64	7/22
Updating : yum-3.4.3-154.el7.centos.1.noarch	8/22
Updating : 32:bind-license-9.9.4-51.el7_4.2.noarch	9/22
Updating : tzdata-2018c-1.el7.noarch	10/22
Updating : kpartx-0.4.9-111.el7_4.2.x86_64	11/22
Cleanup : systemd-219-42.el7_4.4.x86_64	12/22
Cleanup : kmod-20-15.el7_4.6.x86_64	13/22
Cleanup : libdb-utils-5.3.21-20.el7.x86_64	14/22
Cleanup : yum-3.4.3-154.el7.centos.noarch	15/22
Cleanup : 32:bind-license-9.9.4-51.el7_4.1.noarch	16/22
Cleanup : tzdata-2017c-1.el7.noarch	17/22
Cleanup : libdb-5.3.21-20.el7.x86_64	18/22
Cleanup : binutils-2.25.1-32.base.el7_4.1.x86_64	19/22
Cleanup : kmod-libs-20-15.el7_4.6.x86_64	20/22
Cleanup : systemd-libs-219-42.el7_4.4.x86_64	21/22
Cleanup : kpartx-0.4.9-111.el7.x86_64	22/22
Verifying : kmod-20-15.el7_4.7.x86_64	1/22
Verifying : kmod-libs-20-15.el7_4.7.x86_64	2/22
Verifying : libdb-utils-5.3.21-21.el7_4.x86_64	3/22
Verifying : systemd-219-42.el7_4.7.x86_64	4/22
Verifying : kpartx-0.4.9-111.el7_4.2.x86_64	5/22
Verifying : tzdata-2018c-1.el7.noarch	6/22
Verifying : 32:bind-license-9.9.4-51.el7_4.2.noarch	7/22
Verifying : systemd-libs-219-42.el7_4.7.x86_64	8/22
Verifying : binutils-2.25.1-32.base.el7_4.2.x86_64	9/22
Verifying : libdb-5.3.21-21.el7_4.x86_64	10/22
Verifying : yum-3.4.3-154.el7.centos.1.noarch	11/22
Verifying : binutils-2.25.1-32.base.el7_4.1.x86_64	12/22
Verifying : 32:bind-license-9.9.4-51.el7_4.1.noarch	13/22
Verifying : systemd-libs-219-42.el7_4.4.x86_64	14/22
Verifying : kmod-20-15.el7_4.6.x86_64	15/22
Verifying : systemd-219-42.el7_4.4.x86_64	16/22
Verifying : libdb-utils-5.3.21-20.el7.x86_64	17/22
Verifying : kmod-libs-20-15.el7_4.6.x86_64	18/22
Verifying : tzdata-2017c-1.el7.noarch	19/22
Verifying : kpartx-0.4.9-111.el7.x86_64	20/22
Verifying : yum-3.4.3-154.el7.centos.noarch	21/22
Verifying : libdb-5.3.21-20.el7.x86_64	22/22

(continues on next page)

(continued from previous page)

```

Updated:
bind-license.noarch 32:9.9.4-51.el7_4.2
binutils.x86_64 0:2.25.1-32.base.el7_4.2
kmod.x86_64 0:20-15.el7_4.7
kmod-libs.x86_64 0:20-15.el7_4.7
kpartx.x86_64 0:0.4.9-111.el7_4.2
libdbd.x86_64 0:5.3.21-21.el7_4
libdbd-utils.x86_64 0:5.3.21-21.el7_4
systemd.x86_64 0:219-42.el7_4.7
systemd-libs.x86_64 0:219-42.el7_4.7
tzdata.noarch 0:2018c-1.el7
yum.noarch 0:3.4.3-154.el7.centos.1

Complete!
Removing intermediate container bd9bc627aeeb
--> 90814f4b95d5
Step 3/5 : RUN yum install -y https://centos7.iuscommunity.org/ius-release.rpm
--> Running in cea6a40470fa
Loaded plugins: fastestmirror, ovl
Examining /var/tmp/yum-root-Z3I8ac/ius-release.rpm: ius-release-1.0-15.ius.centos7.
→noarch
Marking /var/tmp/yum-root-Z3I8ac/ius-release.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package ius-release.noarch 0:1.0-15.ius.centos7 will be installed
--> Processing Dependency: epel-release = 7 for package: ius-release-1.0-15.ius.
→centos7.noarch
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* extras: fr.mirror.babylon.network
* updates: fr.mirror.babylon.network
--> Running transaction check
--> Package epel-release.noarch 0:7-9 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version            Repository      Size
=====
Installing:
ius-release      noarch    1.0-15.ius.centos7   /ius-release   8.5 k
Installing for dependencies:
epel-release     noarch    7-9                  extras        14 k

Transaction Summary
=====
Install 1 Package (+1 Dependent package)

Total size: 23 k
Total download size: 14 k
Installed size: 33 k
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded

```

(continues on next page)

(continued from previous page)

```
Running transaction
Installing : epel-release-7-9.noarch                                         1/2
Installing : ius-release-1.0-15.ius.centos7.noarch                           2/2
Verifying  : ius-release-1.0-15.ius.centos7.noarch                           1/2
Verifying  : epel-release-7-9.noarch                                         2/2

Installed:
ius-release.noarch 0:1.0-15.ius.centos7

Dependency Installed:
epel-release.noarch 0:7-9

Complete!
Removing intermediate container cea6a40470fa
--> b9963da64678
Step 4/5 : RUN yum install -y python36u python36u-libs python36u-devel python36u-pip
--> Running in f9691783f72c
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: fr.mirror.babylon.network
* extras: fr.mirror.babylon.network
* ius: mirrors.tongji.edu.cn
* updates: fr.mirror.babylon.network
Resolving Dependencies
--> Running transaction check
--> Package python36u.x86_64 0:3.6.4-1.ius.centos7 will be installed
--> Package python36u-devel.x86_64 0:3.6.4-1.ius.centos7 will be installed
--> Package python36u-libs.x86_64 0:3.6.4-1.ius.centos7 will be installed
--> Package python36u-pip.noarch 0:9.0.1-1.ius.centos7 will be installed
--> Processing Dependency: python36u-setuptools for package: python36u-pip-9.0.1-1.
--> ius.centos7.noarch
--> Running transaction check
--> Package python36u-setuptools.noarch 0:36.6.0-1.ius.centos7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package          Arch      Version       Repository
               Size
=====
Installing:
python36u        x86_64   3.6.4-1.ius.centos7    ius      56 k
python36u-devel  x86_64   3.6.4-1.ius.centos7    ius     839 k
python36u-libs   x86_64   3.6.4-1.ius.centos7    ius      8.7 M
python36u-pip    noarch   9.0.1-1.ius.centos7    ius      1.8 M
Installing for dependencies:
python36u-setuptools noarch  36.6.0-1.ius.centos7    ius      587 k

Transaction Summary
=====
Install 4 Packages (+1 Dependent package)

Total download size: 12 M
Installed size: 53 M
```

(continues on next page)

(continued from previous page)

```

Downloading packages:
warning: /var/cache/yum/x86_64/7/ius/packages/python36u-devel-3.6.4-1.ius.centos7.x86_
→64.rpm: Header V4 DSA/SHA1 Signature, key ID 9cd4953f: NOKEY
Public key for python36u-devel-3.6.4-1.ius.centos7.x86_64.rpm is not installed
-----
Total 1.0 MB/s | 12 MB 00:12
Retrieving key from file:///etc/pki/rpm-gpg/IUS-COMMUNITY-GPG-KEY
Importing GPG key 0x9CD4953F:
Userid : "IUS Community Project <coredev@iuscommunity.org>"
Fingerprint: 8b84 6e3a b3fe 6462 74e8 670f da22 1cdf 9cd4 953f
Package : ius-release-1.0-15.ius.centos7.noarch (installed)
From   : /etc/pki/rpm-gpg/IUS-COMMUNITY-GPG-KEY
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : python36u-libs-3.6.4-1.ius.centos7.x86_64 1/5
Installing : python36u-3.6.4-1.ius.centos7.x86_64 2/5
Installing : python36u-setuptools-36.6.0-1.ius.centos7.noarch 3/5
Installing : python36u-pip-9.0.1-1.ius.centos7.noarch 4/5
Installing : python36u-devel-3.6.4-1.ius.centos7.x86_64 5/5
Verifying  : python36u-setuptools-36.6.0-1.ius.centos7.noarch 1/5
Verifying  : python36u-pip-9.0.1-1.ius.centos7.noarch 2/5
Verifying  : python36u-3.6.4-1.ius.centos7.x86_64 3/5
Verifying  : python36u-libs-3.6.4-1.ius.centos7.x86_64 4/5
Verifying  : python36u-devel-3.6.4-1.ius.centos7.x86_64 5/5

Installed:
python36u.x86_64 0:3.6.4-1.ius.centos7
python36u-devel.x86_64 0:3.6.4-1.ius.centos7
python36u-libs.x86_64 0:3.6.4-1.ius.centos7
python36u-pip.noarch 0:9.0.1-1.ius.centos7

Dependency Installed:
python36u-setuptools.noarch 0:36.6.0-1.ius.centos7

Complete!
Removing intermediate container f9691783f72c
--> 2edcf9418ddb
Step 5/5 : RUN yum install -y which
--> Running in b7bf8af2a677
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: mirror.airenetworks.es
* extras: fr.mirror.babylon.network
* ius: mirrors.ircam.fr
* updates: fr.mirror.babylon.network
Resolving Dependencies
--> Running transaction check
--> Package which.x86_64 0:2.20-7.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
=====
```

Package	Arch	Version	Repository	Size
(continues on next page)				

(continued from previous page)

```
=====
Installing:
which           x86_64          2.20-7.el7      base        41 k

Transaction Summary
=====
Install 1 Package

Total download size: 41 k
Installed size: 75 k
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : which-2.20-7.el7.x86_64                               1/1
install-info: No such file or directory for /usr/share/info/which.info.gz
Verifying  : which-2.20-7.el7.x86_64                               1/1

Installed:
which.x86_64 0:2.20-7.el7

Complete!
Removing intermediate container b7bf8af2a677
--> c0efabb4e2cb
Successfully built c0efabb4e2cb
Successfully tagged id3centos7:1
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows_
↳ Docker host. All files and directories added to build context will have '-rwxr-xr-x'
↳ permissions. It is recommended to double check and reset permissions for_
↳ sensitive files and directories.
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
↳ docker images
REPOSITORY          TAG      IMAGE ID      CREATED     SIZE
id3centos7         1        c0efabb4e2cb  54 seconds ago  770MB
ch4messageboardapp_web    latest   a08febb741e4  17 hours ago  782MB
postgres            10.1    b820823c41bd  17 hours ago  290MB
<none>              <none>  62b12eb064b3  17 hours ago  729MB
<none>              <none>  46dc0ae69726  17 hours ago  729MB
<none>              <none>  b940cde74b73  17 hours ago  920MB
<none>              <none>  ad18d8d88ab0  18 hours ago  920MB
<none>              <none>  71e39ba2a7bb  18 hours ago  729MB
<none>              <none>  9fda17d01d46  18 hours ago  729MB
<none>              <none>  326079a0d350  18 hours ago  772MB
<none>              <none>  a617107b453b  18 hours ago  772MB
```

(continues on next page)

(continued from previous page)

<none>	<none>	8fdb1af40b0f	19 hours ago	[link]
centos	7	ff426288ea90	3 weeks ago	[link]
nginx	latest	3f8a4339aadd	5 weeks ago	[link]
python	3.6	c1e459c00dc3	6 weeks ago	[link]
postgres	<none>	ec61d13c8566	7 weeks ago	[link]
docker4w/nsenter-dockerd	latest	cae870735e91	3 months ago	[link]
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>				
<none>				
doc				

24.15.9 docker images

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
<none> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	[link]
<none>				
SIZE				
id3centos7	1	c0efabb4e2cb	54 seconds ago	[link]
<none>				
770MB				
ch4messageboardapp_web	latest	a08febb741e4	17 hours ago	[link]
<none>				
782MB				
postgres	10.1	b820823c41bd	17 hours ago	[link]
<none>				
290MB				
<none>	<none>	62b12eb064b3	17 hours ago	[link]
<none>				
729MB				
<none>	<none>	46dc0ae69726	17 hours ago	[link]
<none>				
920MB				
<none>	<none>	b940cd74b73	17 hours ago	[link]
<none>				
920MB				
<none>	<none>	ad18d8d88ab0	18 hours ago	[link]
<none>				
729MB				
<none>	<none>	71e39ba2a7bb	18 hours ago	[link]
<none>				
729MB				
<none>	<none>	9fdal7d01d46	18 hours ago	[link]
<none>				
772MB				
<none>	<none>	326079a0d350	18 hours ago	[link]
<none>				
772MB				
<none>	<none>	a617107b453b	18 hours ago	[link]
<none>				
729MB				
<none>	<none>	8fdb1af40b0f	19 hours ago	[link]
<none>				
729MB				
centos	7	ff426288ea90	3 weeks ago	[link]
<none>				
207MB				
nginx	latest	3f8a4339aadd	5 weeks ago	[link]
<none>				
108MB				
python	3.6	c1e459c00dc3	6 weeks ago	[link]
<none>				
692MB				
postgres	<none>	ec61d13c8566	7 weeks ago	[link]
<none>				
287MB				

(continues on next page)

(continued from previous page)

docker4w/nsenter-dockerd	latest	cae870735e91	3 months ago	...
	↳ 187 kB			

24.15.10 docker run –name test -it id3centos7:1

24.15.11 Problème avec regex

regex = “*”

```
-----
Failed building wheel for regex
Running setup.py clean for regex
Failed to build regex
Installing collected packages: regex
Running setup.py install for regex ... error
Complete output from command /opt/intranet/intranet/bin/python3.6 -u -c "import_
↳ setuptools, tokenize; __file__='/tmp/pip-build-rrdh2091/regex/setup.py';
↳ f=getattr(tokenize, 'open', open)(__file__);code=f.read().replace('\r\n', '\n');f.
↳ close();exec(compile(code, __file__, 'exec'))" install --record /tmp/pip-fjizm5wj-
↳ record/install-record.txt --single-version-externally-managed --compile --install-
↳ headers /opt/intranet/intranet/include/site/python3.6/regex:
/opt/intranet/intranet/lib/python3.6/site-packages/setuptools/dist.py:355:_
↳ UserWarning: Normalizing '2018.01.10' to '2018.1.10'
normalized_version,
running install
running build
running build_py
creating build
creating build/lib.linux-x86_64-3.6
copying regex_3/regex.py -> build/lib.linux-x86_64-3.6
copying regex_3/_regex_core.py -> build/lib.linux-x86_64-3.6
copying regex_3/test_regex.py -> build/lib.linux-x86_64-3.6
running build_ext
building '_regex' extension
creating build/temp.linux-x86_64-3.6
creating build/temp.linux-x86_64-3.6/regex_3
gcc -pthread -Wno-unused-result -Wsign-compare -DDYNAMIC_ANNOTATIONS_ENABLED=1 -
↳ -DNDEBUG -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-
↳ -strong -param=ssp-buffer-size=4 -frecord-gcc-switches -m64 -mtune=generic -D_GNU_
↳ _SOURCE -fPIC -fwrapv -fPIC -I/usr/include/python3.6m -c regex_3/_regex.c -o build/
↳ temp.linux-x86_64-3.6/regex_3/_regex.o
unable to execute 'gcc': No such file or directory
error: command 'gcc' failed with exit status 1

-----
Command "/opt/intranet/intranet/bin/python3.6 -u -c "import setuptools, tokenize;_
↳ __file__='/tmp/pip-build-rrdh2091/regex/setup.py';f=getattr(tokenize, 'open', open)(__
↳ file__);code=f.read().replace('\r\n', '\n');f.close();exec(compile(code, __file__,
↳ '_exec'))" install --record /tmp/pip-fjizm5wj-record/install-record.txt --single-
↳ version-externally-managed --compile --install-headers /opt/intranet/intranet/
↳ include/site/python3.6/regex" failed with error code 1 in /tmp/pip-build-rrdh2091/
↳ regex/
(intranet) [root@35d914e8c996 intranet]# yum install gcc gcc-devel
```

24.15.12 yum install gcc

```
(intranet) [root@35d914e8c996 intranet]# yum install gcc gcc-devel
```

```
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: mirror.vutbr.cz
* extras: fr.mirror.babylon.network
* ius: mirror.team-cymru.org
* updates: fr.mirror.babylon.network
No package gcc-devel available.
Resolving Dependencies
--> Running transaction check
--> Package gcc.x86_64 0:4.8.5-16.el7_4.1 will be installed
--> Processing Dependency: libgomp = 4.8.5-16.el7_4.1 for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: cpp = 4.8.5-16.el7_4.1 for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: glibc-devel >= 2.2.90-12 for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: libmpfr.so.4()(64bit) for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: libmpc.so.3()(64bit) for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: libgomp.so.1()(64bit) for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Running transaction check
--> Package cpp.x86_64 0:4.8.5-16.el7_4.1 will be installed
--> Package glibc-devel.x86_64 0:2.17-196.el7_4.2 will be installed
--> Processing Dependency: glibc-headers = 2.17-196.el7_4.2 for package: glibc-devel-2.17-196.el7_4.2.x86_64
--> Processing Dependency: glibc-headers for package: glibc-devel-2.17-196.el7_4.2.x86_64
--> Package libgomp.x86_64 0:4.8.5-16.el7_4.1 will be installed
--> Package libmpc.x86_64 0:1.0.1-3.el7 will be installed
--> Package mpfr.x86_64 0:3.1.1-4.el7 will be installed
--> Running transaction check
--> Package glibc-headers.x86_64 0:2.17-196.el7_4.2 will be installed
--> Processing Dependency: kernel-headers >= 2.2.1 for package: glibc-headers-2.17-196.el7_4.2.x86_64
--> Processing Dependency: kernel-headers for package: glibc-headers-2.17-196.el7_4.2.x86_64
--> Running transaction check
--> Package kernel-headers.x86_64 0:3.10.0-693.17.1.el7 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

Package	Arch	Repository
Version		
Size		
Installing:		
gcc	x86_64	updates
4.8.5-16.el7_4.1		
16 M		

(continues on next page)

(continued from previous page)

```

Installing for dependencies:
cpp                                     x86_64
↳ 4.8.5-16.el7_4.1                      updates
↳      5.9 M
glibc-devel                               x86_64
↳ 2.17-196.el7_4.2                      updates
↳      1.1 M
glibc-headers                            x86_64
↳ 2.17-196.el7_4.2                      updates
↳      676 k
kernel-headers                           x86_64
↳ 3.10.0-693.17.1.el7                  updates
↳      6.0 M
libgomp                                  x86_64
↳ 4.8.5-16.el7_4.1                      updates
↳      154 k
libmpc                                    x86_64
↳ 1.0.1-3.el7                           base
↳      51 k
mpfr                                     x86_64
↳ 3.1.1-4.el7                           base
↳      203 k

Transaction Summary
=====
Install 1 Package (+7 Dependent packages)

Total download size: 30 M
Installed size: 60 M
Is this ok [y/d/N]: y
Downloading packages:
(1/8): glibc-headers-2.17-196.el7_4.2.x86_64.rpm
↳ kB 00:00:01                           | 676
(2/8): libgomp-4.8.5-16.el7_4.1.x86_64.rpm
↳ kB 00:00:00                           | 154
(3/8): glibc-devel-2.17-196.el7_4.2.x86_64.rpm
↳ MB 00:00:02                           | 1.1
(4/8): libmpc-1.0.1-3.el7.x86_64.rpm
↳ kB 00:00:00                           | 51
(5/8): mpfr-3.1.1-4.el7.x86_64.rpm
↳ kB 00:00:00                           | 203
(6/8): cpp-4.8.5-16.el7_4.1.x86_64.rpm
↳ MB 00:00:05                           | 5.9
(7/8): kernel-headers-3.10.0-693.17.1.el7.x86_64.rpm
↳ MB 00:00:12                           | 6.0
(8/8): gcc-4.8.5-16.el7_4.1.x86_64.rpm
↳ MB 00:01:13                           | 16
-----
```

(continues on next page)

(continued from previous page)

```
Total                                         ↴
↳ MB 00:01:13                                         ↴
↳ Running transaction check                      ↴
↳ Running transaction test                      ↴
↳ Transaction test succeeded                   ↴
↳ Running transaction                         ↴
↳ Installing : mpfr-3.1.1-4.el7.x86_64       ↴
↳ ↳ 1/8                                           ↴
↳ Installing : libmpc-1.0.1-3.el7.x86_64       ↴
↳ ↳ 2/8                                           ↴
↳ Installing : cpp-4.8.5-16.el7_4.1.x86_64     ↴
↳ ↳ 3/8                                           ↴
↳ Installing : kernel-headers-3.10.0-693.17.1.el7.x86_64
↳ ↳ 4/8                                           ↴
↳ Installing : glibc-headers-2.17-196.el7_4.2.x86_64
↳ ↳ 5/8                                           ↴
↳ Installing : glibc-devel-2.17-196.el7_4.2.x86_64
↳ ↳ 6/8                                           ↴
↳ Installing : libgomp-4.8.5-16.el7_4.1.x86_64
↳ ↳ 7/8                                           ↴
↳ Installing : gcc-4.8.5-16.el7_4.1.x86_64      ↴
↳ ↳ 8/8                                           ↴
↳ Verifying : cpp-4.8.5-16.el7_4.1.x86_64       ↴
↳ ↳ 1/8                                           ↴
↳ Verifying : glibc-devel-2.17-196.el7_4.2.x86_64
↳ ↳ 2/8                                           ↴
↳ Verifying : mpfr-3.1.1-4.el7.x86_64           ↴
↳ ↳ 3/8                                           ↴
↳ Verifying : libgomp-4.8.5-16.el7_4.1.x86_64
↳ ↳ 4/8                                           ↴
↳ Verifying : libmpc-1.0.1-3.el7.x86_64          ↴
↳ ↳ 5/8                                           ↴
↳ Verifying : kernel-headers-3.10.0-693.17.1.el7.x86_64
↳ ↳ 6/8                                           ↴
↳ Verifying : glibc-headers-2.17-196.el7_4.2.x86_64
↳ ↳ 7/8                                           ↴
↳ Verifying : gcc-4.8.5-16.el7_4.1.x86_64        ↴
↳ ↳ 8/8                                           ↴
Installed:
```

(continues on next page)

(continued from previous page)

```
gcc.x86_64 0:4.8.5-16.el7_4.1

Dependency Installed:
cpp.x86_64 0:4.8.5-16.el7_4.1           glibc-devel.x86_64 0:2.17-196.el7_4.2      ↳
↳ glibc-headers.x86_64 0:2.17-196.el7_4.2       kernel-headers.x86_64 0:3.10.0-693.    ↳
↳ 17.1.el7
libgomp.x86_64 0:4.8.5-16.el7_4.1        libmpc.x86_64 0:1.0.1-3.el7      ↳
↳ mpfr.x86_64 0:3.1.1-4.el7

Complete!
```

```
(intranet) [root@35d914e8c996 intranet]# pip install regex
```

```
Collecting regex
Using cached regex-2018.01.10.tar.gz
Building wheels for collected packages: regex
Running setup.py bdist_wheel for regex ... done
Stored in directory: /root/.cache/pip/wheels/6c/44/28/
↳ d58762d1fbdf2e6f6fb00d4fec7d3384ad0ac565b895c044eb
Successfully built regex
Installing collected packages: regex
Successfully installed regex-2018.1.10
```

24.15.13 yum install openldap-devel

```
(intranet) [root@35d914e8c996 intranet]# yum install openldap-devel
```

```
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: fr.mirror.babylon.network
* extras: fr.mirror.babylon.network
* ius: mirrors.tongji.edu.cn
* updates: fr.mirror.babylon.network
Resolving Dependencies
--> Running transaction check
--> Package openldap-devel.x86_64 0:2.4.44-5.el7 will be installed
--> Processing Dependency: cyrus-sasl-devel(x86-64) for package: openldap-devel-2.4.
↳ 44-5.el7.x86_64
--> Running transaction check
--> Package cyrus-sasl-devel.x86_64 0:2.1.26-21.el7 will be installed
--> Processing Dependency: cyrus-sasl(x86-64) = 2.1.26-21.el7 for package: cyrus-sasl-
↳ devel-2.1.26-21.el7.x86_64
--> Running transaction check
--> Package cyrus-sasl.x86_64 0:2.1.26-21.el7 will be installed
--> Processing Dependency: /sbin/service for package: cyrus-sasl-2.1.26-21.el7.x86_64
--> Running transaction check
--> Package initscripts.x86_64 0:9.49.39-1.el7_4.1 will be installed
--> Processing Dependency: sysvinit-tools >= 2.87-5 for package: initscripts-9.49.39-
↳ 1.el7_4.1.x86_64
--> Processing Dependency: iproute for package: initscripts-9.49.39-1.el7_4.1.x86_64
--> Running transaction check
--> Package iproute.x86_64 0:3.10.0-87.el7 will be installed
```

(continues on next page)

(continued from previous page)

```
--> Processing Dependency: libmnl.so.0(LIBMNL_1.0) (64bit) for package: iproute-3.10.0-87.el7.x86_64
--> Processing Dependency: libxtables.so.10() (64bit) for package: iproute-3.10.0-87.el7.x86_64
--> Processing Dependency: libmnl.so.0() (64bit) for package: iproute-3.10.0-87.el7.x86_64
---> Package sysvinit-tools.x86_64 0:2.88-14.dsfc will be installed
--> Running transaction check
---> Package iptables.x86_64 0:1.4.21-18.2.el7_4 will be installed
--> Processing Dependency: libnfnetlink.so.0() (64bit) for package: iptables-1.4.21-18.2.el7_4.x86_64
--> Processing Dependency: libnetfilter_conntrack.so.3() (64bit) for package: iptables-1.4.21-18.2.el7_4.x86_64
---> Package libmnl.x86_64 0:1.0.3-7.el7 will be installed
--> Running transaction check
---> Package libnetfilter_conntrack.x86_64 0:1.0.6-1.el7_3 will be installed
---> Package libnfnetlink.x86_64 0:1.0.1-4.el7 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

Package	Arch	Repository
Version		
Size		
Installing:		
openldap-devel	x86_64	
2.4.44-5.el7	base	
801 k		
Installing for dependencies:		
cyrus-sasl	x86_64	
2.1.26-21.el7	base	
88 k		
cyrus-sasl-devel	x86_64	
2.1.26-21.el7	base	
310 k		
initscripts	x86_64	
9.49.39-1.el7_4.1	updates	
435 k		
iproute	x86_64	
3.10.0-87.el7	base	
651 k		
iptables	x86_64	
1.4.21-18.2.el7_4	updates	
428 k		
libmnl	x86_64	
1.0.3-7.el7	base	
23 k		
libnetfilter_conntrack	x86_64	
1.0.6-1.el7_3	base	
55 k		
libnfnetlink	x86_64	
1.0.1-4.el7	base	
26 k		
sysvinit-tools	x86_64	
2.88-14.dsfc	base	
63 k		

(continues on next page)

(continued from previous page)

```
Transaction Summary
=====
Install 1 Package (+9 Dependent packages)

Total download size: 2.8 M
Installed size: 9.5 M
Is this ok [y/d/N]: y
Downloading packages:
(1/10): cyrus-sasl-2.1.26-21.el7.x86_64.rpm           | 88 kB  00:00:00
(2/10): cyrus-sasl-devel-2.1.26-21.el7.x86_64.rpm      | 310 kB  00:00:00
(3/10): libmnl-1.0.3-7.el7.x86_64.rpm                  | 23 kB   00:00:00
(4/10): initscripts-9.49.39-1.el7_4.1.x86_64.rpm        | 435 kB  00:00:00
(5/10): libnetfilter_conntrack-1.0.6-1.el7_3.x86_64.rpm | 55 kB   00:00:00
(6/10): libnfnetwork-1.0.1-4.el7.x86_64.rpm            | 26 kB   00:00:00
(7/10): iptables-1.4.21-18.2.el7_4.x86_64.rpm          | 428 kB  00:00:01
(8/10): sysvinit-tools-2.88-14.dsfc.el7.x86_64.rpm     | 63 kB   00:00:00
(9/10): openldap-devel-2.4.44-5.el7.x86_64.rpm          | 801 kB  00:00:00
(10/10): iproute-3.10.0-87.el7.x86_64.rpm              | 651 kB  00:00:01
-----
-----
Total                                         1.2 MB/s | 2.8 kB
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : libnfnetwork-1.0.1-4.el7.x86_64           | 1/10
Installing : libmnl-1.0.3-7.el7.x86_64                  | 2/10
Installing : libnetfilter_conntrack-1.0.6-1.el7_3.x86_64 | 3/10
```

(continues on next page)

(continued from previous page)

```

Installing : iptables-1.4.21-18.2.el7_4.x86_64
→          4/10
Installing : iproute-3.10.0-87.el7.x86_64
→          5/10
Installing : sysvinit-tools-2.88-14.dsfc.el7.x86_64
→          6/10
Installing : initscripts-9.49.39-1.el7_4.1.x86_64
→          7/10
Installing : cyrus-sasl-2.1.26-21.el7.x86_64
→          8/10
Installing : cyrus-sasl-devel-2.1.26-21.el7.x86_64
→          9/10
Installing : openldap-devel-2.4.44-5.el7.x86_64
→          10/10
Verifying  : iptables-1.4.21-18.2.el7_4.x86_64
→          1/10
Verifying  : libmnl-1.0.3-7.el7.x86_64
→          2/10
Verifying  : iproute-3.10.0-87.el7.x86_64
→          3/10
Verifying  : initscripts-9.49.39-1.el7_4.1.x86_64
→          4/10
Verifying  : cyrus-sasl-devel-2.1.26-21.el7.x86_64
→          5/10
Verifying  : libnfnetwork-1.0.1-4.el7.x86_64
→          6/10
Verifying  : sysvinit-tools-2.88-14.dsfc.el7.x86_64
→          7/10
Verifying  : libnetfilter_conntrack-1.0.6-1.el7_3.x86_64
→          8/10
Verifying  : openldap-devel-2.4.44-5.el7.x86_64
→          9/10
Verifying  : cyrus-sasl-2.1.26-21.el7.x86_64
→          10/10

Installed:
openldap-devel.x86_64 0:2.4.44-5.el7

Dependency Installed:
cyrus-sasl.x86_64 0:2.1.26-21.el7           cyrus-sasl-devel.x86_64 0:2.1.26-21.el7
→      initscripts.x86_64 0:9.49.39-1.el7_4.1           iproute.x86_64 (continues on next page)
→      87.el7

```

(continued from previous page)

```
iptables.x86_64 0:1.4.21-18.2.el7_4           libmnl.x86_64 0:1.0.3-7.el7      ↵
↳     libnetfilter_conntrack.x86_64 0:1.0.6-1.el7_3    libnfnetlink.x86_64 0:1.0.   ↵
↳1-4.el7
sysvinit-tools.x86_64 0:2.88-14.dsfc.el7

Complete!
```

24.15.14 pip install pyldap

```
(intranet) [root@35d914e8c996 intranet]# pip install pyldap
```

```
Collecting pyldap
Using cached pyldap-2.4.45.tar.gz
Requirement already satisfied: setuptools in ./intranet/lib/python3.6/site-packages
↳ (from pyldap)
Building wheels for collected packages: pyldap
Running setup.py bdist_wheel for pyldap ... done
Stored in directory: /root/.cache/pip/wheels/0c/a3/42/
↳ e6127de64a53567a11c4e3ee5991547cb8f5a3241d2d67947e
Successfully built pyldap
Installing collected packages: pyldap
Successfully installed pyldap-2.4.45
```

24.15.15 Nouveau fichier Dockerfile

24.15.15.1 Dockerfile

```
# Use an official centos7 image
FROM centos:7

RUN yum update -y \
    && yum install -y https://centos7.iuscommunity.org/ius-release.rpm \
    && yum install -y python36u python36u-libs python36u-devel python36u-pip \
    && yum install -y which gcc \ # we need regex and pyldap
    && yum install -y openldap-devel # we need pyldap
```

24.15.15.2 which python3.6

```
[root@5a070209b99d /]# which python3.6
```

```
/usr/bin/python3.6
```

24.15.15.3 python3.6 -m pip install pipenv

```
python3.6 -m pip install pipenv
```

```

Collecting pipenv
  Downloading pipenv-9.0.3.tar.gz (3.9MB)
    100% | 3.9MB 336kB/s
Collecting virtualenv (from pipenv)
  Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
    100% | 1.8MB 602kB/s
Collecting pew>=0.1.26 (from pipenv)
  Downloading pew-1.1.2-py2.py3-none-any.whl
Requirement already satisfied: pip>=9.0.1 in /usr/lib/python3.6/site-packages (from
→pipenv)
Collecting requests>2.18.0 (from pipenv)
  Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
    100% | 92kB 2.2MB/s
Collecting flake8>=3.0.0 (from pipenv)
  Downloading flake8-3.5.0-py2.py3-none-any.whl (69kB)
    100% | 71kB 1.8MB/s
Collecting urllib3>=1.21.1 (from pipenv)
  Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
    100% | 133kB 1.8MB/s
Requirement already satisfied: setuptools>=17.1 in /usr/lib/python3.6/site-packages
→(from pew>=0.1.26->pipenv)
Collecting virtualenv-clone>=0.2.5 (from pew>=0.1.26->pipenv)
  Downloading virtualenv-clone-0.2.6.tar.gz
Collecting certifi>=2017.4.17 (from requests>2.18.0->pipenv)
  Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
    100% | 153kB 982kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests>2.18.0->pipenv)
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% | 143kB 1.8MB/s
Collecting idna<2.7,>=2.5 (from requests>2.18.0->pipenv)
  Downloading idna-2.6-py2.py3-none-any.whl (56kB)
    100% | 61kB 900kB/s
Collecting mccabe<0.7.0,>=0.6.0 (from flake8>=3.0.0->pipenv)
  Downloading mccabe-0.6.1-py2.py3-none-any.whl
Collecting pycodestyle<2.4.0,>=2.0.0 (from flake8>=3.0.0->pipenv)
  Downloading pycodestyle-2.3.1-py2.py3-none-any.whl (45kB)
    100% | 51kB 2.3MB/s
Collecting pyflakes<1.7.0,>=1.5.0 (from flake8>=3.0.0->pipenv)
  Downloading pyflakes-1.6.0-py2.py3-none-any.whl (227kB)
    100% | 235kB 2.2MB/s
Installing collected packages: virtualenv, virtualenv-clone, pew, urllib3, certifi,
→chardet, idna, requests, mccabe, pycodestyle, pyflakes, flake8, pipenv
  Running setup.py install for virtualenv-clone ... done
  Running setup.py install for pipenv ... done
Successfully installed certifi-2018.1.18 chardet-3.0.4 flake8-3.5.0 idna-2.6 mccabe-0.
→6.1 pew-1.1.2 pipenv-9.0.3 pycodestyle-2.3.1 pyflakes-1.6.0 requests-2.18.4 urllib3-
→1.22 virtualenv-15.1.0 virtualenv-clone-0.2.6

```

24.15.16 Nouveau Dockerfile

24.15.16.1 Dockerfile

```

# Use an official centos7 image
FROM centos:7

```

(continues on next page)

(continued from previous page)

```
RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8

# gcc because we need regex and pyldap
# openldap-devel because we need pyldap
RUN yum update -y \
    && yum install -y https://centos7.iuscommunity.org/ius-release.rpm \
    && yum install -y python36u python36u-libs python36u-devel python36u-pip \
    && yum install -y which gcc \
    && yum install -y openldap-devel

RUN python3.6 -m pip install pipenv
```

24.15.16.2 docker build -t id3centos7:0.1.1 .

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:0.1.1 .
```

```
Sending build context to Docker daemon 90.11kB
Step 1/5 : FROM centos:7
--> ff426288ea90
Step 2/5 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.
→ UTF-8
--> Running in b90f824550e7
Removing intermediate container b90f824550e7
--> b7dac1f044e3
Step 3/5 : ENV LANG fr_FR.utf8
--> Running in 107f8edaf492
Removing intermediate container 107f8edaf492
--> e28a88050b8f
Step 4/5 : RUN yum update -y      && yum install -y https://centos7.iuscommunity.org/
→ ius-release.rpm      && yum install -y python36u python36u-libs python36u-devel
→ python36u-pip      && yum install -y which gcc      && yum install -y openldap-devel
--> Running in 531a6dc0ab1
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
* base: centos.quelquesmots.fr
* extras: ftp.ciril.fr
* updates: centos.quelquesmots.fr
Resolving Dependencies
--> Running transaction check
--> Package bind-license.noarch 32:9.9.4-51.el7_4.1 will be updated
--> Package bind-license.noarch 32:9.9.4-51.el7_4.2 will be an update
--> Package binutils.x86_64 0:2.25.1-32.base.el7_4.1 will be updated
--> Package binutils.x86_64 0:2.25.1-32.base.el7_4.2 will be an update
--> Package kmod.x86_64 0:20-15.el7_4.6 will be updated
--> Package kmod.x86_64 0:20-15.el7_4.7 will be an update
--> Package kmod-libs.x86_64 0:20-15.el7_4.6 will be updated
--> Package kmod-libs.x86_64 0:20-15.el7_4.7 will be an update
--> Package kpartx.x86_64 0:0.4.9-111.el7 will be updated
--> Package kpartx.x86_64 0:0.4.9-111.el7_4.2 will be an update
--> Package libdb.x86_64 0:5.3.21-20.el7 will be updated
--> Package libdb.x86_64 0:5.3.21-21.el7_4 will be an update
--> Package libdb-utils.x86_64 0:5.3.21-20.el7 will be updated
```

(continues on next page)

(continued from previous page)

```
----> Package libdb-utils.x86_64 0:5.3.21-21.el7_4 will be an update
----> Package systemd.x86_64 0:219-42.el7_4.4 will be updated
----> Package systemd.x86_64 0:219-42.el7_4.7 will be an update
----> Package systemd-libs.x86_64 0:219-42.el7_4.4 will be updated
----> Package systemd-libs.x86_64 0:219-42.el7_4.7 will be an update
----> Package tzdata.noarch 0:2017c-1.el7 will be updated
----> Package tzdata.noarch 0:2018c-1.el7 will be an update
----> Package yum.noarch 0:3.4.3-154.el7.centos will be updated
----> Package yum.noarch 0:3.4.3-154.el7.centos.1 will be an update
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package           Arch      Version            Repository      Size
=====
Upgrading:
bind-license     noarch   32:9.9.4-51.el7_4.2    updates        84 k
binutils          x86_64   2.25.1-32.base.el7_4.2   updates       5.4 M
kmod              x86_64   20-15.el7_4.7          updates      121 k
kmod-libs         x86_64   20-15.el7_4.7          updates      50 k
kpartx            x86_64   0.4.9-111.el7_4.2    updates      73 k
libdb             x86_64   5.3.21-21.el7_4        updates     719 k
libdb-utils       x86_64   5.3.21-21.el7_4        updates     132 k
systemd           x86_64   219-42.el7_4.7        updates      5.2 M
systemd-libs      x86_64   219-42.el7_4.7        updates     376 k
tzdata            noarch   2018c-1.el7          updates     479 k
yum               noarch   3.4.3-154.el7.centos.1  updates     1.2 M
```

Transaction Summary

```
=====
Upgrade 11 Packages
```

Total download size: 14 M

Downloading packages:

```
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
warning: /var/cache/yum/x86_64/7/updates/packages/bind-license-9.9.4-51.el7_4.2.
  ↳noarch.rpm: Header V3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEY
Public key for bind-license-9.9.4-51.el7_4.2.noarch.rpm is not installed
```

```
=====
Total                                         1.5 MB/s | 14 MB 00:09
```

Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

Importing GPG key 0xF4A80EB5:

```
Userid      : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
```

```
Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
```

```
Package     : centos-release-7-4.1708.el7.centos.x86_64 (@CentOS)
```

```
From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
```

Running transaction check

Running transaction test

Transaction test succeeded

Running transaction

Updating	:	libdb-5.3.21-21.el7_4.x86_64	1/22
----------	---	------------------------------	------

Updating	:	binutils-2.25.1-32.base.el7_4.2.x86_64	2/22
----------	---	--	------

Updating	:	kmod-20-15.el7_4.7.x86_64	3/22
----------	---	---------------------------	------

Updating	:	systemd-libs-219-42.el7_4.7.x86_64	4/22
----------	---	------------------------------------	------

Updating	:	kmod-libs-20-15.el7_4.7.x86_64	5/22
----------	---	--------------------------------	------

Updating	:	systemd-219-42.el7_4.7.x86_64	6/22
----------	---	-------------------------------	------

(continues on next page)

(continued from previous page)

Updating	:	libdb-utils-5.3.21-21.el7_4.x86_64	7/22
Updating	:	yum-3.4.3-154.el7.centos.1.noarch	8/22
Updating	:	32:bind-license-9.9.4-51.el7_4.2.noarch	9/22
Updating	:	tzdata-2018c-1.el7.noarch	10/22
Updating	:	kpartx-0.4.9-111.el7_4.2.x86_64	11/22
Cleanup	:	systemd-219-42.el7_4.4.x86_64	12/22
Cleanup	:	kmod-20-15.el7_4.6.x86_64	13/22
Cleanup	:	libdb-utils-5.3.21-20.el7.x86_64	14/22
Cleanup	:	yum-3.4.3-154.el7.centos.noarch	15/22
Cleanup	:	32:bind-license-9.9.4-51.el7_4.1.noarch	16/22
Cleanup	:	tzdata-2017c-1.el7.noarch	17/22
Cleanup	:	libdb-5.3.21-20.el7.x86_64	18/22
Cleanup	:	binutils-2.25.1-32.base.el7_4.1.x86_64	19/22
Cleanup	:	kmod-libs-20-15.el7_4.6.x86_64	20/22
Cleanup	:	systemd-libs-219-42.el7_4.4.x86_64	21/22
Cleanup	:	kpartx-0.4.9-111.el7.x86_64	22/22
Verifying	:	kmod-20-15.el7_4.7.x86_64	1/22
Verifying	:	kmod-libs-20-15.el7_4.7.x86_64	2/22
Verifying	:	libdb-utils-5.3.21-21.el7_4.x86_64	3/22
Verifying	:	systemd-219-42.el7_4.7.x86_64	4/22
Verifying	:	kpartx-0.4.9-111.el7_4.2.x86_64	5/22
Verifying	:	tzdata-2018c-1.el7.noarch	6/22
Verifying	:	32:bind-license-9.9.4-51.el7_4.2.noarch	7/22
Verifying	:	systemd-libs-219-42.el7_4.7.x86_64	8/22
Verifying	:	binutils-2.25.1-32.base.el7_4.2.x86_64	9/22
Verifying	:	libdb-5.3.21-21.el7_4.x86_64	10/22
Verifying	:	yum-3.4.3-154.el7.centos.1.noarch	11/22
Verifying	:	binutils-2.25.1-32.base.el7_4.1.x86_64	12/22
Verifying	:	32:bind-license-9.9.4-51.el7_4.1.noarch	13/22
Verifying	:	systemd-libs-219-42.el7_4.4.x86_64	14/22
Verifying	:	kmod-20-15.el7_4.6.x86_64	15/22
Verifying	:	systemd-219-42.el7_4.4.x86_64	16/22
Verifying	:	libdb-utils-5.3.21-20.el7.x86_64	17/22
Verifying	:	kmod-libs-20-15.el7_4.6.x86_64	18/22
Verifying	:	tzdata-2017c-1.el7.noarch	19/22
Verifying	:	kpartx-0.4.9-111.el7.x86_64	20/22
Verifying	:	yum-3.4.3-154.el7.centos.noarch	21/22
Verifying	:	libdb-5.3.21-20.el7.x86_64	22/22
 Updated:			
bind-license.noarch 32:9.9.4-51.el7_4.2			
binutils.x86_64 0:2.25.1-32.base.el7_4.2			
kmod.x86_64 0:20-15.el7_4.7			
kmod-libs.x86_64 0:20-15.el7_4.7			
kpartx.x86_64 0:0.4.9-111.el7_4.2			
libdb.x86_64 0:5.3.21-21.el7_4			
libdb-utils.x86_64 0:5.3.21-21.el7_4			
systemd.x86_64 0:219-42.el7_4.7			
systemd-libs.x86_64 0:219-42.el7_4.7			
tzdata.noarch 0:2018c-1.el7			
yum.noarch 0:3.4.3-154.el7.centos.1			
 Complete!			
Loaded plugins: fastestmirror, ovl			
Examining /var/tmp/yum-root-CU9Amb/ius-release.rpm: ius-release-1.0-15.ius.centos7.			
→noarch			
Marking /var/tmp/yum-root-CU9Amb/ius-release.rpm to be installed			

(continues on next page)

(continued from previous page)

```

Resolving Dependencies
--> Running transaction check
--> Package ius-release.noarch 0:1.0-15.ius.centos7 will be installed
--> Processing Dependency: epel-release = 7 for package: ius-release-1.0-15.ius.
→centos7.noarch
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* extras: ftp.ciril.fr
* updates: centos.quelquesmots.fr
--> Running transaction check
--> Package epel-release.noarch 0:7-9 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package           Arch      Version            Repository      Size
=====
Installing:
ius-release      noarch    1.0-15.ius.centos7   /ius-release   8.5 k
Installing for dependencies:
epel-release     noarch    7-9                  extras        14 k

Transaction Summary
=====
Install 1 Package (+1 Dependent package)

Total size: 23 k
Total download size: 14 k
Installed size: 33 k
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : epel-release-7-9.noarch                         1/2
  Installing : ius-release-1.0-15.ius.centos7.noarch          2/2
  Verifying   : ius-release-1.0-15.ius.centos7.noarch          1/2
  Verifying   : epel-release-7-9.noarch                         2/2

Installed:
ius-release.noarch 0:1.0-15.ius.centos7

Dependency Installed:
epel-release.noarch 0:7-9

Complete!
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: fr.mirror.babylon.network
* extras: ftp.ciril.fr
* ius: mirrors.ircam.fr
* updates: centos.quelquesmots.fr
Resolving Dependencies
--> Running transaction check
--> Package python36u.x86_64 0:3.6.4-1.ius.centos7 will be installed

```

(continues on next page)

(continued from previous page)

```
---> Package python36u-devel.x86_64 0:3.6.4-1.ius.centos7 will be installed
---> Package python36u-libs.x86_64 0:3.6.4-1.ius.centos7 will be installed
---> Package python36u-pip.noarch 0:9.0.1-1.ius.centos7 will be installed
--> Processing Dependency: python36u-setuptools for package: python36u-pip-9.0.1-1.
    ↪ius.centos7.noarch
--> Running transaction check
---> Package python36u-setuptools.noarch 0:36.6.0-1.ius.centos7 will be installed
--> Finished Dependency Resolution
```

Dependencies Resolved

```
=====
Package           Arch      Version       Repository
               ↪
               ↪           Size
=====
Installing:
python36u          x86_64   3.6.4-1.ius.centos7   ius      56 k
python36u-devel     x86_64   3.6.4-1.ius.centos7   ius     839 k
python36u-libs       x86_64   3.6.4-1.ius.centos7   ius      8.7 M
python36u-pip        noarch   9.0.1-1.ius.centos7   ius      1.8 M
Installing for dependencies:
python36u-setuptools noarch   36.6.0-1.ius.centos7   ius      587 k
```

Transaction Summary

```
=====
Install 4 Packages (+1 Dependent package)

Total download size: 12 M
Installed size: 53 M
Downloading packages:
warning: /var/cache/yum/x86_64/7/ius/packages/python36u-setuptools-36.6.0-1.ius.
    ↪centos7.noarch.rpm: Header V4 DSA/SHA1 Signature, key ID 9cd4953f: NOKEY
Public key for python36u-setuptools-36.6.0-1.ius.centos7.noarch.rpm is not installed
=====
Total                                         634 kB/s | 12 MB  00:19
Retrieving key from file:///etc/pki/rpm-gpg/IUS-COMMUNITY-GPG-KEY
Importing GPG key 0x9CD4953F:
Userid      : "IUS Community Project <coredev@iuscommunity.org>"
Fingerprint: 8b84 6e3a b3fe 6462 74e8 670f da22 1cdf 9cd4 953f
Package     : ius-release-1.0-15.ius.centos7.noarch (installed)
From       : /etc/pki/rpm-gpg/IUS-COMMUNITY-GPG-KEY
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : python36u-libs-3.6.4-1.ius.centos7.x86_64                         1/5
Installing : python36u-3.6.4-1.ius.centos7.x86_64                           2/5
Installing : python36u-setuptools-36.6.0-1.ius.centos7.noarch                   3/5
Installing : python36u-pip-9.0.1-1.ius.centos7.noarch                   4/5
Installing : python36u-devel-3.6.4-1.ius.centos7.x86_64                      5/5
Verifying  : python36u-setuptools-36.6.0-1.ius.centos7.noarch                   1/5
Verifying  : python36u-pip-9.0.1-1.ius.centos7.noarch                   2/5
Verifying  : python36u-3.6.4-1.ius.centos7.x86_64                      3/5
Verifying  : python36u-libs-3.6.4-1.ius.centos7.x86_64                     4/5
Verifying  : python36u-devel-3.6.4-1.ius.centos7.x86_64                     5/5
```

(continues on next page)

(continued from previous page)

```

Installed:
python36u.x86_64 0:3.6.4-1.ius.centos7
python36u-devel.x86_64 0:3.6.4-1.ius.centos7
python36u-libs.x86_64 0:3.6.4-1.ius.centos7
python36u-pip.noarch 0:9.0.1-1.ius.centos7

Dependency Installed:
python36u-setuptools.noarch 0:36.6.0-1.ius.centos7

Complete!
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: fr.mirror.babylon.network
* extras: ftp.ciril.fr
* ius: mirrors.ircam.fr
* updates: centos.quelquesmots.fr
Resolving Dependencies
--> Running transaction check
----> Package gcc.x86_64 0:4.8.5-16.el7_4.1 will be installed
--> Processing Dependency: libgomp = 4.8.5-16.el7_4.1 for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: cpp = 4.8.5-16.el7_4.1 for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: glibc-devel >= 2.2.90-12 for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: libmpfr.so.4()(64bit) for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: libmpc.so.3()(64bit) for package: gcc-4.8.5-16.el7_4.1.x86_64
--> Processing Dependency: libgomp.so.1()(64bit) for package: gcc-4.8.5-16.el7_4.1.x86_64
----> Package which.x86_64 0:2.20-7.el7 will be installed
--> Running transaction check
----> Package cpp.x86_64 0:4.8.5-16.el7_4.1 will be installed
----> Package glibc-devel.x86_64 0:2.17-196.el7_4.2 will be installed
--> Processing Dependency: glibc-headers = 2.17-196.el7_4.2 for package: glibc-devel-2.17-196.el7_4.2.x86_64
--> Processing Dependency: glibc-headers for package: glibc-devel-2.17-196.el7_4.2.x86_64
----> Package libgomp.x86_64 0:4.8.5-16.el7_4.1 will be installed
----> Package libmpc.x86_64 0:1.0.1-3.el7 will be installed
----> Package mpfr.x86_64 0:3.1.1-4.el7 will be installed
--> Running transaction check
----> Package glibc-headers.x86_64 0:2.17-196.el7_4.2 will be installed
--> Processing Dependency: kernel-headers >= 2.2.1 for package: glibc-headers-2.17-196.el7_4.2.x86_64
--> Processing Dependency: kernel-headers for package: glibc-headers-2.17-196.el7_4.2.x86_64
--> Running transaction check
----> Package kernel-headers.x86_64 0:3.10.0-693.17.1.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved
=====

```

Package	Arch	Version	Repository	Size
(continues on next page)				

(continued from previous page)

```
=====
Installing:
gcc           x86_64      4.8.5-16.el7_4.1          updates      16 M
which         x86_64      2.20-7.el7                base        41 k
=====
Installing for dependencies:
cpp            x86_64      4.8.5-16.el7_4.1          updates      5.9 M
glibc-devel    x86_64      2.17-196.el7_4.2        updates      1.1 M
glibc-headers  x86_64      2.17-196.el7_4.2        updates      676 k
kernel-headers x86_64      3.10.0-693.17.1.el7    updates      6.0 M
libgomp        x86_64      4.8.5-16.el7_4.1          updates      154 k
libmpc         x86_64      1.0.1-3.el7                base        51 k
mpfr          x86_64      3.1.1-4.el7                base        203 k
=====
Transaction Summary
=====
Install 2 Packages (+7 Dependent packages)

Total download size: 30 M
Installed size: 60 M
Downloading packages:
-----
Total                                         1.3 MB/s | 30 MB 00:23
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : mpfr-3.1.1-4.el7.x86_64          1/9
Installing : libmpc-1.0.1-3.el7.x86_64        2/9
Installing : cpp-4.8.5-16.el7_4.1.x86_64      3/9
Installing : kernel-headers-3.10.0-693.17.1.el7.x86_64 4/9
Installing : glibc-headers-2.17-196.el7_4.2.x86_64 5/9
Installing : glibc-devel-2.17-196.el7_4.2.x86_64 6/9
Installing : libgomp-4.8.5-16.el7_4.1.x86_64    7/9
Installing : gcc-4.8.5-16.el7_4.1.x86_64       8/9
Installing : which-2.20-7.el7.x86_64          9/9
install-info: No such file or directory for /usr/share/info/which.info.gz
Verifying   : cpp-4.8.5-16.el7_4.1.x86_64      1/9
Verifying   : glibc-devel-2.17-196.el7_4.2.x86_64 2/9
Verifying   : which-2.20-7.el7.x86_64          3/9
Verifying   : mpfr-3.1.1-4.el7.x86_64          4/9
Verifying   : libgomp-4.8.5-16.el7_4.1.x86_64 5/9
Verifying   : libmpc-1.0.1-3.el7.x86_64        6/9
Verifying   : kernel-headers-3.10.0-693.17.1.el7.x86_64 7/9
Verifying   : glibc-headers-2.17-196.el7_4.2.x86_64 8/9
Verifying   : gcc-4.8.5-16.el7_4.1.x86_64       9/9
-----
Installed:
gcc.x86_64 0:4.8.5-16.el7_4.1                  which.x86_64 0:2.20-7.el7
-----
Dependency Installed:
cpp.x86_64 0:4.8.5-16.el7_4.1
glibc-devel.x86_64 0:2.17-196.el7_4.2
glibc-headers.x86_64 0:2.17-196.el7_4.2
kernel-headers.x86_64 0:3.10.0-693.17.1.el7
libgomp.x86_64 0:4.8.5-16.el7_4.1
libmpc.x86_64 0:1.0.1-3.el7
mpfr.x86_64 0:3.1.1-4.el7

```

(continues on next page)

(continued from previous page)

```

Complete!
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
* base: centos.quelquesmots.fr
* epel: fr.mirror.babylon.network
* extras: ftp.ciril.fr
* ius: mirrors.ircam.fr
* updates: centos.quelquesmots.fr
Resolving Dependencies
--> Running transaction check
---> Package openldap-devel.x86_64 0:2.4.44-5.el7 will be installed
--> Processing Dependency: cyrus-sasl-devel(x86-64) for package: openldap-devel-2.4.
  ↳ 44-5.el7.x86_64
--> Running transaction check
---> Package cyrus-sasl-devel.x86_64 0:2.1.26-21.el7 will be installed
--> Processing Dependency: cyrus-sasl(x86-64) = 2.1.26-21.el7 for package: cyrus-sasl-
  ↳ devel-2.1.26-21.el7.x86_64
--> Running transaction check
---> Package cyrus-sasl.x86_64 0:2.1.26-21.el7 will be installed
--> Processing Dependency: /sbin/service for package: cyrus-sasl-2.1.26-21.el7.x86_64
--> Running transaction check
---> Package initscripts.x86_64 0:9.49.39-1.el7_4.1 will be installed
--> Processing Dependency: sysvinit-tools >= 2.87-5 for package: initscripts-9.49.39-
  ↳ 1.el7_4.1.x86_64
--> Processing Dependency: iproute for package: initscripts-9.49.39-1.el7_4.1.x86_64
--> Running transaction check
---> Package iproute.x86_64 0:3.10.0-87.el7 will be installed
--> Processing Dependency: libmnl.so.0(LIBMNL_1.0)(64bit) for package: iproute-3.10.0-
  ↳ 87.el7.x86_64
--> Processing Dependency: libxtables.so.10()(64bit) for package: iproute-3.10.0-87.
  ↳ el7.x86_64
--> Processing Dependency: libmnl.so.0()(64bit) for package: iproute-3.10.0-87.el7.
  ↳ x86_64
--> Package sysvinit-tools.x86_64 0:2.88-14.dsfc.el7 will be installed
--> Running transaction check
---> Package iptables.x86_64 0:1.4.21-18.2.el7_4 will be installed
--> Processing Dependency: libnfnetlink.so.0()(64bit) for package: iptables-1.4.21-18.
  ↳ 2.el7_4.x86_64
--> Processing Dependency: libnetfilter_conntrack.so.3()(64bit) for package: iptables-
  ↳ 1.4.21-18.2.el7_4.x86_64
--> Package libmnl.x86_64 0:1.0.3-7.el7 will be installed
--> Running transaction check
---> Package libnetfilter_conntrack.x86_64 0:1.0.6-1.el7_3 will be installed
---> Package libnfnetlink.x86_64 0:1.0.1-4.el7 will be installed
--> Finished Dependency Resolution

```

Dependencies Resolved

Package	Arch	Version	Repository	Size
<hr/>				
Installing:				
openldap-devel	x86_64	2.4.44-5.el7	base	801 k
<hr/>				
Installing for dependencies:				
cyrus-sasl	x86_64	2.1.26-21.el7	base	88 k
cyrus-sasl-devel	x86_64	2.1.26-21.el7	base	310 k

(continues on next page)

(continued from previous page)

initscripts	x86_64	9.49.39-1.el7_4.1	updates	435 k
iproute	x86_64	3.10.0-87.el7	base	651 k
iptables	x86_64	1.4.21-18.2.el7_4	updates	428 k
libmnl	x86_64	1.0.3-7.el7	base	23 k
libnetfilter_conntrack	x86_64	1.0.6-1.el7_3	base	55 k
libnfnetlink	x86_64	1.0.1-4.el7	base	26 k
sysvinit-tools	x86_64	2.88-14.dsfc.el7	base	63 k
 Transaction Summary =====				
Install 1 Package (+9 Dependent packages)				
Total download size: 2.8 M				
Installed size: 9.5 M				
Downloading packages:				

Total			1.2 MB/s 2.8 MB 00:02	
Running transaction check				
Running transaction test				
Transaction test succeeded				
Running transaction				
Installing : libnfnetlink-1.0.1-4.el7.x86_64			1/10	
Installing : libmnl-1.0.3-7.el7.x86_64			2/10	
Installing : libnetfilter_conntrack-1.0.6-1.el7_3.x86_64			3/10	
Installing : iptables-1.4.21-18.2.el7_4.x86_64			4/10	
Installing : iproute-3.10.0-87.el7.x86_64			5/10	
Installing : sysvinit-tools-2.88-14.dsfc.el7.x86_64			6/10	
Installing : initscripts-9.49.39-1.el7_4.1.x86_64			7/10	
Installing : cyrus-sasl-2.1.26-21.el7.x86_64			8/10	
Installing : cyrus-sasl-devel-2.1.26-21.el7.x86_64			9/10	
Installing : openldap-devel-2.4.44-5.el7.x86_64			10/10	
Verifying : iptables-1.4.21-18.2.el7_4.x86_64			1/10	
Verifying : libmnl-1.0.3-7.el7.x86_64			2/10	
Verifying : iproute-3.10.0-87.el7.x86_64			3/10	
Verifying : initscripts-9.49.39-1.el7_4.1.x86_64			4/10	
Verifying : cyrus-sasl-devel-2.1.26-21.el7.x86_64			5/10	
Verifying : libnfnetlink-1.0.1-4.el7.x86_64			6/10	
Verifying : sysvinit-tools-2.88-14.dsfc.el7.x86_64			7/10	
Verifying : libnetfilter_conntrack-1.0.6-1.el7_3.x86_64			8/10	
Verifying : openldap-devel-2.4.44-5.el7.x86_64			9/10	
Verifying : cyrus-sasl-2.1.26-21.el7.x86_64			10/10	
 Installed:				
openldap-devel.x86_64 0:2.4.44-5.el7				
 Dependency Installed:				
cyrus-sasl.x86_64 0:2.1.26-21.el7				
cyrus-sasl-devel.x86_64 0:2.1.26-21.el7				
initscripts.x86_64 0:9.49.39-1.el7_4.1				
iproute.x86_64 0:3.10.0-87.el7				
iptables.x86_64 0:1.4.21-18.2.el7_4				
libmnl.x86_64 0:1.0.3-7.el7				
libnetfilter_conntrack.x86_64 0:1.0.6-1.el7_3				
libnfnetlink.x86_64 0:1.0.1-4.el7				
sysvinit-tools.x86_64 0:2.88-14.dsfc.el7				
 Complete!				

(continues on next page)

(continued from previous page)

```

Removing intermediate container 531a6dcb0ab1
--> 0cfdf4200049
Step 5/5 : RUN python3.6 -m pip install pipenv
--> Running in 222c51c8c187
Collecting pipenv
  Downloading pipenv-9.0.3.tar.gz (3.9MB)
    Collecting virtualenv (from pipenv)
      Downloading virtualenv-15.1.0-py2.py3-none-any.whl (1.8MB)
        Collecting pew>=0.1.26 (from pipenv)
          Downloading pew-1.1.2-py2.py3-none-any.whl
            Requirement already satisfied: pip>=9.0.1 in /usr/lib/python3.6/site-packages (from
             ↵pipenv)
          Collecting requests>2.18.0 (from pipenv)
            Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
          Collecting flake8>=3.0.0 (from pipenv)
            Downloading flake8-3.5.0-py2.py3-none-any.whl (69kB)
          Collecting urllib3>=1.21.1 (from pipenv)
            Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
            Requirement already satisfied: setuptools>=17.1 in /usr/lib/python3.6/site-packages
             ↵(from pew>=0.1.26->pipenv)
          Collecting virtualenv-clone>=0.2.5 (from pew>=0.1.26->pipenv)
            Downloading virtualenv-clone-0.2.6.tar.gz
          Collecting certifi>=2017.4.17 (from requests>2.18.0->pipenv)
            Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
          Collecting chardet<3.1.0,>=3.0.2 (from requests>2.18.0->pipenv)
            Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
          Collecting idna<2.7,>=2.5 (from requests>2.18.0->pipenv)
            Downloading idna-2.6-py2.py3-none-any.whl (56kB)
          Collecting pycodestyle<2.4.0,>=2.0.0 (from flake8>=3.0.0->pipenv)
            Downloading pycodestyle-2.3.1-py2.py3-none-any.whl (45kB)
          Collecting mccabe<0.7.0,>=0.6.0 (from flake8>=3.0.0->pipenv)
            Downloading mccabe-0.6.1-py2.py3-none-any.whl
          Collecting pyflakes<1.7.0,>=1.5.0 (from flake8>=3.0.0->pipenv)
            Downloading pyflakes-1.6.0-py2.py3-none-any.whl (227kB)
        Installing collected packages: virtualenv, virtualenv-clone, pew, certifi, chardet,
         ↵idna, urllib3, requests, pycodestyle, mccabe, pyflakes, flake8, pipenv
      Running setup.py install for virtualenv-clone: started
      Running setup.py install for virtualenv-clone: finished with status 'done'
      Running setup.py install for pipenv: started
      Running setup.py install for pipenv: finished with status 'done'
      Successfully installed certifi-2018.1.18 chardet-3.0.4 flake8-3.5.0 idna-2.6 mccabe-0.
       ↵6.1 pew-1.1.2 pipenv-9.0.3 pycodestyle-2.3.1 pyflakes-1.6.0 requests-2.18.4 urllib3-
       ↵1.22 virtualenv-15.1.0 virtualenv-clone-0.2.6
    Removing intermediate container 222c51c8c187
--> 9965dbca3f49
  Successfully built 9965dbca3f49
  Successfully tagged id3centos7:0.1.1
  SECURITY WARNING: You are building a Docker image from Windows against a non-Windows
   ↵Docker host. All files and directories added to build context will have '-rwxr-xr-x
   ↵' permissions. It is recommended to double check and reset permissions for
   ↵sensitive files and directories.
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>

```

24.15.17 Nouveau fichier Dockerfile

24.15.17.1 Dockerfile

```
# Use an official centos7 image
FROM centos:7

RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8

# gcc because we need regex and pyldap
# openldap-devel because we need pyldap
RUN yum update -y \
    && yum install -y https://centos7.iuscommunity.org/ius-release.rpm \
    && yum install -y python36u python36u-libs python36u-devel python36u-pip \
    && yum install -y which gcc \
    && yum install -y openldap-devel

RUN python3.6 -m pip install pipenv

WORKDIR /opt/intranet
COPY Pipfile /opt/intranet/
```

24.15.17.2 Construction de l'image docker build -t id3centos7:0.1.2 .

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:0.1.2 .
```

```
Sending build context to Docker daemon 195.1kB
Step 1/7 : FROM centos:7
--> ff426288ea90
Step 2/7 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
--> Using cache
--> b7dac1f044e3
Step 3/7 : ENV LANG fr_FR.utf8
--> Using cache
--> e28a88050b8f
Step 4/7 : RUN yum update -y && yum install -y https://centos7.iuscommunity.org/ius-release.rpm && yum install -y python36u python36u-libs python36u-devel python36u-pip && yum install -y which gcc && yum install -y openldap-devel
--> Using cache
--> 0cfdf4200049
Step 5/7 : RUN python3.6 -m pip install pipenv
--> Using cache
--> 9965dbc3f49
Step 6/7 : WORKDIR /opt/intranet
Removing intermediate container ffc087754a0c
--> aecca04b51f8
Step 7/7 : COPY Pipfile /opt/intranet/
--> e126ba1ca5f5
Successfully built e126ba1ca5f5
Successfully tagged id3centos7:0.1.2
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.
```

(continued from previous page)

24.15.17.3 docker run --name id3centos7.1.2 -it id3centos7:0.1.2

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker run --name id3centos7.1.2 -it id3centos7:0.1.2
```

```
[root@8586df0dcb8e intranet]# pwd
/opt/intranet
```

```
[root@8586df0dcb8e intranet]# ls -als
```

```
total 12
4 drwxr-xr-x 1 root root 4096 févr.  2 13:43 .
4 drwxr-xr-x 1 root root 4096 févr.  2 13:43 ..
4 -rwxr-xr-x 1 root root  910 févr.  2 11:23 Pipfile
```

Problème : la commande pipenv

24.15.18 Nouveau dockerfile

24.15.18.1 Dockerfile

```
# Use an official centos7 image
FROM centos:7

RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8

# gcc because we need regex and pyldap
# openldap-devel because we need pyldap
RUN yum update -y \
    && yum install -y https://centos7.iuscommunity.org/ius-release.rpm \
    && yum install -y python36u python36u-libs python36u-devel python36u-pip \
    && yum install -y which gcc \
    && yum install -y openldap-devel

RUN python3.6 -m pip install pipenv

WORKDIR /opt/intranet

# copy the Pipfile to the working directory
COPY Pipfile /opt/intranet/
# https://docs.pipenv.org/advanced/
# This is useful for Docker containers, and deployment infrastructure (e.g. Heroku)
# does this
RUN pipenv install
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:0.1.3 .
```

```
Sending build context to Docker daemon 198.1kB
Step 1/8 : FROM centos:7
--> ff426288ea90
Step 2/8 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.
--> UTF-8
--> Using cache
--> b7dac1f044e3
Step 3/8 : ENV LANG fr_FR.utf8
--> Using cache
--> e28a88050b8f
Step 4/8 : RUN yum update -y && yum install -y https://centos7.iuscommunity.org/
--> ius-release.rpm && yum install -y python36u python36u-libs python36u-devel
--> python36u-pip && yum install -y which gcc && yum install -y openldap-devel
--> Using cache
--> 0cfdf4200049
Step 5/8 : RUN python3.6 -m pip install pipenv
--> Using cache
--> 9965dbc3f49
Step 6/8 : WORKDIR /opt/intranet
--> Using cache
--> aecca04b51f8
Step 7/8 : COPY Pipfile /opt/intranet/
--> Using cache
--> 188cff4aa6e9
Step 8/8 : RUN pipenv install
--> Running in cdc65d965685
Creating a virtualenv for this project...
Using base prefix '/usr'
New python executable in /root/.local/share/virtualenvs/intranet-6TUV_xiL/bin/python3.
--> 6
Also creating executable in /root/.local/share/virtualenvs/intranet-6TUV_xiL/bin/
--> python
Installing setuptools, pip, wheel...done.

Virtualenv location: /root/.local/share/virtualenvs/intranet-6TUV_xiL
Pipfile.lock not found, creating...
Locking [dev-packages] dependencies...
Locking [packages] dependencies...
Updated Pipfile.lock (326c76)!
Installing dependencies from Pipfile.lock (326c76)...
To activate this project's virtualenv, run the following:
$ pipenv shell
Removing intermediate container cdc65d965685
--> 179eac6f62c1
Successfully built 179eac6f62c1
Successfully tagged id3centos7:0.1.3
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows
Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions.
It is recommended to double check and reset permissions for sensitive files and
--> directories.
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
```

24.15.19 Nouveau fichier Dockerfile

24.15.19.1 Dockerfile

```
# Use an official centos7 image
FROM centos:7

RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.UTF-8
ENV LANG fr_FR.utf8

# gcc because we need regex and pyldap
# openldap-devel because we need pyldap
RUN yum update -y \
    && yum install -y https://centos7.iuscommunity.org/ius-release.rpm \
    && yum install -y python36u python36u-libs python36u-devel python36u-pip \
    && yum install -y which gcc \
    && yum install -y openldap-devel

RUN python3.6 -m pip install pipenv

WORKDIR /opt/intranet

# copy the Pipfile to the working directory
ONBUILD COPY Pipfile /opt/intranet/
# https://docs.pipenv.org/advanced/
# https://github.com/pypa/pipenv/issues/1385
# This is useful for Docker containers, and deployment infrastructure (e.g. Heroku_
→does this)
ONBUILD RUN pipenv install --system
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:0.1.4 .
```

```
Sending build context to Docker daemon 201.2kB
Step 1/8 : FROM centos:7
--> ff426288ea90
Step 2/8 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.
→UTF-8
--> Using cache
--> b7dac1f044e3
Step 3/8 : ENV LANG fr_FR.utf8
--> Using cache
--> e28a88050b8f
Step 4/8 : RUN yum update -y      && yum install -y https://centos7.iuscommunity.org/
→ius-release.rpm      && yum install -y python36u python36u-libs python36u-devel_
→python36u-pip      && yum install -y which gcc      && yum install -y openldap-devel
--> Using cache
--> 0cfdf4200049
Step 5/8 : RUN python3.6 -m pip install pipenv
--> Using cache
--> 9965dbc3f49
Step 6/8 : WORKDIR /opt/intranet
--> Using cache
--> aecca04b51f8
Step 7/8 : ONBUILD COPY Pipfile /opt/intranet/
--> Running in 0d30cd780e8c
```

(continues on next page)

(continued from previous page)

```

Removing intermediate container 0d30cd780e8c
--> c4a15216b54b
Step 8/8 : ONBUILD RUN pipenv install --system
--> Running in 9bb757ba3d15
Removing intermediate container 9bb757ba3d15
--> 237ec53f0462
Successfully built 237ec53f0462
Successfully tagged id3centos7:0.1.4
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows_
↳ Docker host. All files and directories added to build context will have '-rwxr-xr-x'_
↳ permissions. It is recommended to double check and reset permissions for_
↳ sensitive files and directories.

```

24.15.20 Nouveau fichier Dockerfile

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:0.1.6 .
```

```

Sending build context to Docker daemon 240.6kB
Step 1/8 : FROM centos:7
--> ff426288ea90
Step 2/8 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.
↳ UTF-8
--> Using cache
--> b7dac1f044e3
Step 3/8 : ENV LANG fr_FR.utf8
--> Using cache
--> e28a88050b8f
Step 4/8 : RUN yum update -y && yum install -y https://centos7.iuscommunity.org/
↳ ius-release.rpm && yum install -y python36u python36u-libs python36u-devel
↳ python36u-pip && yum install -y which gcc && yum install -y openldap-devel
--> Using cache
--> 0cfdf4200049
Step 5/8 : RUN python3.6 -m pip install pipenv
--> Using cache
--> 9965dbca3f49
Step 6/8 : WORKDIR /opt/intranet
--> Using cache
--> aecca04b51f8
Step 7/8 : COPY requirements.txt /opt/intranet/
--> 8ae3427dbfca
Step 8/8 : RUN pip install -r requirements.txt
--> Running in 555693a8d7bb
/bin/sh: pip: command not found
The command '/bin/sh -c pip install -r requirements.txt' returned a non-zero code: 127
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
→ docker build -t id3centos7:0.1.6 .
Sending build context to Docker daemon 240.6kB
Step 1/7 : FROM centos:7
--> ff426288ea90
Step 2/7 : RUN localedef -i fr_FR -c -f UTF-8 -A /usr/share/locale/locale.alias fr_FR.
↳ UTF-8
--> Using cache
--> b7dac1f044e3

```

(continues on next page)

(continued from previous page)

```

Step 3/7 : ENV LANG fr_FR.utf8
--> Using cache
--> e28a88050b8f
Step 4/7 : RUN yum update -y      && yum install -y https://centos7.iuscommunity.org/
  ↵ius-release.rpm      && yum install -y python36u python36u-libs python36u-devel
  ↵python36u-pip      && yum install -y which gcc      && yum install -y openldap-devel
--> Using cache
--> 0cfdf4200049
Step 5/7 : WORKDIR /opt/intranet
Removing intermediate container 2af4e31fb8ed
--> 7fb09cc14c29
Step 6/7 : COPY requirements.txt /opt/intranet/
--> eecebec115f4
Step 7/7 : RUN python3.6 -m pip install -r requirements.txt
--> Running in 8400df97d2aa
Collecting arrow==0.12.1 (from -r requirements.txt (line 1))
  Downloading arrow-0.12.1.tar.gz (65kB)
Collecting babel==2.5.3 (from -r requirements.txt (line 2))
  Downloading Babel-2.5.3-py2.py3-none-any.whl (6.8MB)
Collecting certifi==2018.1.18 (from -r requirements.txt (line 3))
  Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
Collecting chardet==3.0.4 (from -r requirements.txt (line 4))
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
Collecting dateparser==0.6.0 (from -r requirements.txt (line 5))
  Downloading dateparser-0.6.0-py2.py3-none-any.whl (68kB)
Collecting diff-match-patch==20121119 (from -r requirements.txt (line 6))
  Downloading diff-match-patch-20121119.tar.gz (54kB)
Collecting django==2.0.2 (from -r requirements.txt (line 7))
  Downloading Django-2.0.2-py3-none-any.whl (7.1MB)
Collecting django-ajax-selects==1.7.0 (from -r requirements.txt (line 8))
  Downloading django_ajax_selects-1.7.0-py3-none-any.whl
Collecting django-autocomplete-light==3.2.10 (from -r requirements.txt (line 9))
  Downloading django-autocomplete-light-3.2.10.tar.gz (428kB)
Collecting django-bootstrap4==0.0.5 (from -r requirements.txt (line 10))
  Downloading django-bootstrap4-0.0.5.tar.gz
Collecting django-braces==1.12.0 (from -r requirements.txt (line 11))
  Downloading django_braces-1.12.0-py2.py3-none-any.whl
Collecting django-countries==5.1.1 (from -r requirements.txt (line 12))
  Downloading django_countries-5.1.1-py2.py3-none-any.whl (682kB)
Collecting django-crispy-forms==1.7.0 (from -r requirements.txt (line 13))
  Downloading django_crispy_forms-1.7.0-py2.py3-none-any.whl (104kB)
Collecting django-embed-video==1.1.2 (from -r requirements.txt (line 14))
  Downloading django-embed-video-1.1.2.tar.gz
Collecting django-environ==0.4.4 (from -r requirements.txt (line 15))
  Downloading django_environ-0.4.4-py2.py3-none-any.whl
Collecting django-extended-choices==1.2 (from -r requirements.txt (line 16))
  Downloading django_extended_choices-1.2-py2.py3-none-any.whl
Collecting django-extensions==1.9.9 (from -r requirements.txt (line 17))
  Downloading django_extensions-1.9.9-py2.py3-none-any.whl (213kB)
Collecting django-import-export==0.7.0 (from -r requirements.txt (line 18))
  Downloading django_import_export-0.7.0-py2.py3-none-any.whl (72kB)
Collecting django-localflavor==2.0 (from -r requirements.txt (line 19))
  Downloading django_localflavor-2.0-py2.py3-none-any.whl (2.4MB)
Collecting django-money==0.12.3 (from -r requirements.txt (line 20))
  Downloading django_money-0.12.3-py2.py3-none-any.whl
Collecting django-phonenumber-field==2.0.0 (from -r requirements.txt (line 21))
  Downloading django_phonenumber_field-2.0.0.tar.gz

```

(continues on next page)

(continued from previous page)

```

Collecting djangorestframework==3.7.7 (from -r requirements.txt (line 22))
  Downloading djangorestframework-3.7.7-py2.py3-none-any.whl (1.1MB)
Collecting et-xmlfile==1.0.1 (from -r requirements.txt (line 23))
  Downloading et_xmlfile-1.0.1.tar.gz
Collecting ftfy==5.3.0 (from -r requirements.txt (line 24))
  Downloading ftfy-5.3.0.tar.gz (53kB)
Collecting future==0.16.0 (from -r requirements.txt (line 25))
  Downloading future-0.16.0.tar.gz (824kB)
Collecting idna==2.6 (from -r requirements.txt (line 26))
  Downloading idna-2.6-py2.py3-none-any.whl (56kB)
Collecting jdcal==1.3 (from -r requirements.txt (line 27))
  Downloading jdcal-1.3.tar.gz
Collecting odfpy==1.3.6 (from -r requirements.txt (line 28))
  Downloading odfpy-1.3.6.tar.gz (691kB)
Collecting openpyxl==2.5.0 (from -r requirements.txt (line 29))
  Downloading openpyxl-2.5.0.tar.gz (169kB)
Collecting pendulum==1.4.0 (from -r requirements.txt (line 30))
  Downloading pendulum-1.4.0-cp36-cp36m-manylinux1_x86_64.whl (127kB)
Collecting phonenumberslite==8.8.10 (from -r requirements.txt (line 31))
  Downloading phonenumberslite-8.8.10-py2.py3-none-any.whl (429kB)
Collecting pillow==5.0.0 (from -r requirements.txt (line 32))
  Downloading Pillow-5.0.0-cp36-cp36m-manylinux1_x86_64.whl (5.9MB)
Collecting prettytable==0.7.2 (from -r requirements.txt (line 33))
  Downloading prettytable-0.7.2.zip
Collecting psycopg2==2.7.3.2 (from -r requirements.txt (line 34))
  Downloading psycopg2-2.7.3.2-cp36-cp36m-manylinux1_x86_64.whl (2.7MB)
Collecting py-moneyed==0.7.0 (from -r requirements.txt (line 35))
  Downloading py_moneyed-0.7.0-py3-none-any.whl
Collecting python-dateutil==2.6.1 (from -r requirements.txt (line 36))
  Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
Collecting pytz==2017.3 (from -r requirements.txt (line 37))
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
Collecting pytzdata==2018.3 (from -r requirements.txt (line 38))
  Downloading pytzdata-2018.3-py2.py3-none-any.whl (492kB)
Collecting pyyaml==3.12 (from -r requirements.txt (line 39))
  Downloading PyYAML-3.12.tar.gz (253kB)
Collecting regex==2018.1.10 (from -r requirements.txt (line 40))
  Downloading regex-2018.01.10.tar.gz (612kB)
Collecting requests==2.18.4 (from -r requirements.txt (line 41))
  Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
Collecting ruamel.yaml==0.15.35 (from -r requirements.txt (line 42))
  Downloading ruamel.yaml-0.15.35-cp36-cp36m-manylinux1_x86_64.whl (558kB)
Collecting six==1.11.0 (from -r requirements.txt (line 43))
  Downloading six-1.11.0-py2.py3-none-any.whl
Collecting sorl-thumbnail==12.4.1 (from -r requirements.txt (line 44))
  Downloading sorl_thumbnail-12.4.1-py2.py3-none-any.whl (44kB)
Collecting sqlanydb==1.0.9 (from -r requirements.txt (line 45))
  Downloading sqlanydb-1.0.9.tar.gz
Collecting tablib==0.12.1 (from -r requirements.txt (line 46))
  Downloading tablib-0.12.1.tar.gz (63kB)
Collecting typing==3.6.4 (from -r requirements.txt (line 47))
  Downloading typing-3.6.4-py3-none-any.whl
Collecting tzlocal==1.5.1 (from -r requirements.txt (line 48))
  Downloading tzlocal-1.5.1.tar.gz
Collecting unicodecsv==0.14.1 (from -r requirements.txt (line 49))
  Downloading unicodecsv-0.14.1.tar.gz
Collecting urllib3==1.22 (from -r requirements.txt (line 50))

```

(continues on next page)

(continued from previous page)

```

Downloading urllib3==1.22-py2.py3-none-any.whl (132kB)
Collecting wcwidth==0.1.7 (from -r requirements.txt (line 51))
  Downloading wcwidth-0.1.7-py2.py3-none-any.whl
Collecting xlrd==1.1.0 (from -r requirements.txt (line 52))
  Downloading xlrd-1.1.0-py2.py3-none-any.whl (108kB)
Collecting xlwt==1.3.0 (from -r requirements.txt (line 53))
  Downloading xlwt-1.3.0-py2.py3-none-any.whl (99kB)
Requirement already satisfied: setuptools in /usr/lib/python3.6/site-packages (from
  ↵django-money==0.12.3->-r requirements.txt (line 20))
Installing collected packages: six, python-dateutil, arrow, pytz, babel, certifi,_
  ↵chardet, regex, ruamel.yaml, tzlocal, dateparser, diff-match-patch, django, django-
  ↵ajax-selects, django-autocomplete-light, django-bootstrap4, django-braces, django-
  ↵countries, django-crispy-forms, idna, urllib3, requests, django-embed-video, django-
  ↵environs, future, django-extended-choices, typing, django-extensions, odfpy, jdcal,_
  ↵et-xmlfile, openpyxl, unicodecsv, xlrd, xlwt, pyyaml, tablib, django-import-export,_
  ↵django-localflavor, py-moneyed, django-money, phonenumberslite, django-phonenum-
  ↵ber-field, djangorestframework, wcwidth, ftfy, pytzdata, pendulum, pillow, prettytable,_
  ↵psycopg2, sorl-thumbnail, sqlanydb
Running setup.py install for arrow: started
Running setup.py install for arrow: finished with status 'done'
Running setup.py install for regex: started
Running setup.py install for regex: finished with status 'done'
Running setup.py install for tzlocal: started
Running setup.py install for tzlocal: finished with status 'done'
Running setup.py install for diff-match-patch: started
Running setup.py install for diff-match-patch: finished with status 'done'
Running setup.py install for django-autocomplete-light: started
Running setup.py install for django-autocomplete-light: finished with status 'done'
Running setup.py install for django-bootstrap4: started
Running setup.py install for django-bootstrap4: finished with status 'done'
Running setup.py install for django-embed-video: started
Running setup.py install for django-embed-video: finished with status 'done'
Running setup.py install for future: started
Running setup.py install for future: finished with status 'done'
Running setup.py install for odfpy: started
Running setup.py install for odfpy: finished with status 'done'
Running setup.py install for jdcal: started
Running setup.py install for jdcal: finished with status 'done'
Running setup.py install for et-xmlfile: started
Running setup.py install for et-xmlfile: finished with status 'done'
Running setup.py install for openpyxl: started
Running setup.py install for openpyxl: finished with status 'done'
Running setup.py install for unicodecsv: started
Running setup.py install for unicodecsv: finished with status 'done'
Running setup.py install for pyyaml: started
Running setup.py install for pyyaml: finished with status 'done'
Running setup.py install for tablib: started
Running setup.py install for tablib: finished with status 'done'
Running setup.py install for django-phonenumberslite: started
Running setup.py install for django-phonenumberslite: finished with status 'done'
Running setup.py install for ftfy: started
Running setup.py install for ftfy: finished with status 'done'
Running setup.py install for prettytable: started
Running setup.py install for prettytable: finished with status 'done'
Running setup.py install for sqlanydb: started
Running setup.py install for sqlanydb: finished with status 'done'
Successfully installed arrow==0.12.1 babel==2.5.3 certifi==2018.1.18 chardet==3.0.4
  ↵dateparser==0.6.0 diff-match-patch==20121119 django==2.0.2 django-ajax-selects==1.1.2 (continues on next page)
  ↵django-autocomplete-light==3.2.10 django-bootstrap4==0.0.5 django-braces==1.12.0
  ↵django-countries==5.1.1 django-crispy-forms==1.7.0 django-embed-video==1.1.2 django-
  ↵environs==0.4.4 django-extended-choices==1.2 django-extensions==1.9.9 django-import-
  ↵export==0.7.0 django-localflavor==2.0 django-money==0.12.3 django-phonenumberslite==2.
  ↵0.0 djangorestframework==3.7.7 et-xmlfile==1.0.1 ftfy==5.3.0 future==0.16.0 idna==2.6
  ↵jdcal==1.3 odfpy==1.3.6 openpyxl==2.5.0 pendulum==1.4.0 phonenumberslite==8.8.10 pillow-
  ↵5.2.0 pytzdata==2.0.0 prettytable==1.0.0 pyyaml==3.14 sqlanydb==1.0.0

```

(continued from previous page)

```
Removing intermediate container 8400df97d2aa
--> bf91ebbc265a
Successfully built bf91ebbc265a
Successfully tagged id3centos7:0.1.6
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows_
↳ Docker host. All files and directories added to build context will have '-rwxr-xr-x
↳' permissions. It is recommended to double check and reset permissions for
↳ sensitive files and directories.
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\centos7>
```

24.16 Tutoriel Docker et Postgresql

See also:

- *Images PostgreSQL*
- <https://wsvincent.com/django-docker-postgresql/>
- <https://github.com/wsvincent/djangoforbeginners>
- <https://stackoverflow.com/questions/29852583/docker-compose-accessing-postgres-shell-psql>
- *Tutoriel Docker et Postgresql*
- *Mardi 30 janvier 2018 : écriture des fichiers Dockerfile et docker-compose.yml*
- *Images PostgreSQL*
- <https://github.com/slardiere>
- <https://docs.postgresql.fr/10/charset.html>

Contents

- *Tutoriel Docker et Postgresql*
 - *Modèle de fichier docker-compose.yml*
 - *docker-compose up*
 - *docker-compose run postgres psql -h postgres -U postgres*
 - *docker-compose down*
 - *docker-compose build*
 - *docker-compose up*
 - *docker-compose exec -u postgres db psql*
 - *docker ps*
 - *docker exec -it d205b9239366 bash*
 - *Mardi 30 janvier 2018*
 - * *docker-compose.yml*
 - * *docker volume ls*
 - * *docker volume inspect postgresql_volume_intranet*

```

* docker exec -it 47501acda106 bash
* psql -U postgres
* l (liste des bases de données)
* CREATE USER id3admin WITH PASSWORD 'id338';
* CREATE DATABASE db_id3_intranet WITH OWNER = id3admin ENCODING = 'UTF8' CONNECTION LIMIT = -1;
* l
* docker-compose run db env
* docker-compose config
- Import de la base de données
- Mercredi 31 janvier 2018 : export/import d'une base de données PostgreSQL (tutoriel PostgreSQL)
* pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet
* Entête de db.dump
* Expérience substitution de db_id3_save à db_id3_intranet
* psql -U postgres -f db.dump.sql
* docker-compose stop
* docker-compose build
- CREATE DATABASE db_id3_save WITH TEMPLATE = template0 ENCODING = 'UTF8'
LC_COLLATE = 'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

```

24.16.1 Modèle de fichier docker-compose.yml

```

version: "3"

services:
  postgres:
    image: postgres:9.5

```

24.16.2 docker-compose up

```

Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>
→docker-compose up

```

```

WARNING: The Docker Engine you're using is running in swarm mode.

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All ↴
containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

Creating network "postgresql_default" with the default driver
Pulling postgres (postgres:10)...

```

(continues on next page)

The screenshot shows a Stack Overflow post titled "A year on and things are even better: \$ docker-compose exec -u postgres postgres psql". The post was made by "Rory Hart" on Sep 6 '17 at 10:23. It has 13k views, 1 upvote, 33 comments, and 20 answers. The post content includes a code block for a minimal `docker-compose.yml` file:

```
version: "3"
services:
  postgres:
    image: postgres:9.5
```

Below the code block, it says "You bring your services up with `docker-compose up`". The post also includes an answer from "Tate Thurston" with 923 upvotes, 7 comments, and 11 answers. The answer provides a command to run PostgreSQL in a Docker container.

Fig. 13: stack_overflow_postgres.png

(continued from previous page)

```

10: Pulling from library/postgres
Digest: sha256:3f4441460029e12905a5d447a3549ae2ac13323d045391b0cb0cf8b48ea17463
Status: Downloaded newer image for postgres:10
Creating postgresql_postgres_1 ... done
Attaching to postgresql_postgres_1
postgres_1 | The files belonging to this database system will be owned by user
             ↵"postgres".
postgres_1 | This user must also own the server process.
postgres_1 |
postgres_1 | The database cluster will be initialized with locale "en_US.utf8".
postgres_1 | The default database encoding has accordingly been set to "UTF8".
postgres_1 | The default text search configuration will be set to "english".
postgres_1 |
postgres_1 | Data page checksums are disabled.
postgres_1 |
postgres_1 | fixing permissions on existing directory /var/lib/postgresql/data ... ok
postgres_1 | creating subdirectories ... ok
postgres_1 | selecting default max_connections ... 100
postgres_1 | selecting default shared_buffers ... 128MB
postgres_1 | selecting dynamic shared memory implementation ... posix
postgres_1 | creating configuration files ... ok
postgres_1 | running bootstrap script ... ok
postgres_1 | performing post-bootstrap initialization ... ok
postgres_1 | syncing data to disk ...
postgres_1 | WARNING: enabling "trust" authentication for local connections
postgres_1 | You can change this by editing pg_hba.conf or using the option -A, or
postgres_1 | --auth-local and --auth-host, the next time you run initdb.
postgres_1 | ok
postgres_1 |
postgres_1 | Success. You can now start the database server using:
postgres_1 |
postgres_1 |     pg_ctl -D /var/lib/postgresql/data -l logfile start
postgres_1 |
postgres_1 | ****
postgres_1 | WARNING: No password has been set for the database.
postgres_1 |         This will allow anyone with access to the
postgres_1 |         Postgres port to access your database. In
postgres_1 |         Docker's default configuration, this is
postgres_1 |         effectively any other container on the same
postgres_1 |         system.
postgres_1 |
postgres_1 |         Use "-e POSTGRES_PASSWORD=password" to set
postgres_1 |         it in "docker run".
postgres_1 | ****
postgres_1 | waiting for server to start....2018-01-22 11:51:28.410 UTC [37] LOG: ↵
postgres_1 |   listening on IPv4 address "127.0.0.1", port 5432
postgres_1 | 2018-01-22 11:51:28.410 UTC [37] LOG:  could not bind IPv6 address "::1"
postgres_1 |   :: Cannot assign requested address
postgres_1 | 2018-01-22 11:51:28.410 UTC [37] HINT:  Is another postmaster already ↵
postgres_1 |   running on port 5432? If not, wait a few seconds and retry.
postgres_1 | 2018-01-22 11:51:28.510 UTC [37] LOG:  listening on Unix socket "/var/
postgres_1 |   run/postgresql/.s.PGSQL.5432"
postgres_1 | 2018-01-22 11:51:28.712 UTC [38] LOG:  database system was shut down at ↵
postgres_1 |   2018-01-22 11:51:26 UTC
postgres_1 | 2018-01-22 11:51:28.780 UTC [37] LOG:  database system is ready to ↵
postgres_1 |   accept connections
postgres_1 | done

```

(continues on next page)

(continued from previous page)

```

postgres_1 | server started
postgres_1 | ALTER ROLE
postgres_1 |
postgres_1 |
postgres_1 | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.
→d/*
postgres_1 |
postgres_1 | 2018-01-22 11:51:28.985 UTC [37] LOG:  received fast shutdown request
postgres_1 | waiting for server to shut down....2018-01-22 11:51:29.037 UTC [37] LOG:
→LOG:  aborting any active transactions
postgres_1 | 2018-01-22 11:51:29.042 UTC [37] LOG:  worker process: logical
→replication launcher (PID 44) exited with exit code 1
postgres_1 | 2018-01-22 11:51:29.042 UTC [39] LOG:  shutting down
postgres_1 | 2018-01-22 11:51:29.405 UTC [37] LOG:  database system is shut down
postgres_1 | done
postgres_1 | server stopped
postgres_1 |
postgres_1 | PostgreSQL init process complete; ready for start up.
postgres_1 |
postgres_1 | 2018-01-22 11:51:29.565 UTC [1] LOG:  listening on IPv4 address "0.0.0.0
→", port 5432
postgres_1 | 2018-01-22 11:51:29.565 UTC [1] LOG:  listening on IPv6 address "::",
→port 5432
postgres_1 | 2018-01-22 11:51:29.665 UTC [1] LOG:  listening on Unix socket "/var/
→run/postgresql/.s.PGSQL.5432"
postgres_1 | 2018-01-22 11:51:29.825 UTC [55] LOG:  database system was shut down at
→2018-01-22 11:51:29 UTC
postgres_1 | 2018-01-22 11:51:29.878 UTC [1] LOG:  database system is ready to
→accept connections

```

24.16.3 docker-compose run postgres psql -h postgres -U postgres

```

Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>
→docker-compose run postgres psql -h postgres -U postgres

```

```

psql (10.1)
Type "help" for help.

postgres=#

```

```
postgres=# help
```

```

You are using psql, the command-line interface to PostgreSQL.
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit
postgres=#

```

24.16.4 docker-compose down

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>
→docker-compose down
```

```
Stopping postgresql_postgres_1 ... done
Removing postgresql_postgres_run_2 ... done
Removing postgresql_postgres_run_1 ... done
Removing postgresql_postgres_1     ... done
Removing network postgresql_default
```

```
postgres_1 | 2018-01-22 11:51:29.565 UTC [1] LOG:  listening on IPv4 address "0.0.0.0"
←", port 5432
postgres_1 | 2018-01-22 11:51:29.565 UTC [1] LOG:  listening on IPv6 address "::",
←port 5432
postgres_1 | 2018-01-22 11:51:29.665 UTC [1] LOG:  listening on Unix socket "/var/
←run/postgresql/.s.PGSQL.5432"
postgres_1 | 2018-01-22 11:51:29.825 UTC [55] LOG:  database system was shut down at
←2018-01-22 11:51:29 UTC
postgres_1 | 2018-01-22 11:51:29.878 UTC [1] LOG:  database system is ready to
←accept connections
postgres_1 | 2018-01-22 11:56:12.567 UTC [66] FATAL:  database "test" does not exist
postgres_1 | 2018-01-22 12:08:39.698 UTC [1] LOG:  received smart shutdown request
postgres_1 | 2018-01-22 12:08:39.749 UTC [1] LOG:  worker process: logical
←replication launcher (PID 61) exited with exit code 1
postgres_1 | 2018-01-22 12:08:39.750 UTC [56] LOG:  shutting down
postgres_1 | 2018-01-22 12:08:39.965 UTC [1] LOG:  database system is shut down
postgresql_postgres_1 exited with code 0
```

```
version: "3"

services:
  db:
    image: postgres:10.1
    volumes:
      - postgres_data:/var/lib/postgresql/data/
```

24.16.5 docker-compose build

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>
→docker-compose build
```

```
db uses an image, skipping
```

24.16.6 docker-compose up

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>
→docker-compose up
```

```
WARNING: The Docker Engine you're using is running in swarm mode.
```

(continues on next page)

(continued from previous page)

Compose does not use swarm mode to deploy services to multiple nodes in a swarm. All ↵ containers will be scheduled on the current node.

To deploy your application across the swarm, use `docker stack deploy`.

```
Creating network "postgresql_default" with the default driver
Creating volume "postgresql_postgres_data" with default driver
Creating postgresql_db_1 ... done
Attaching to postgresql_db_1
db_1  | The files belonging to this database system will be owned by user "postgres".
db_1  | This user must also own the server process.
db_1  |
db_1  | The database cluster will be initialized with locale "en_US.utf8".
db_1  | The default database encoding has accordingly been set to "UTF8".
db_1  | The default text search configuration will be set to "english".
db_1  |
db_1  | Data page checksums are disabled.
db_1  |
db_1  | fixing permissions on existing directory /var/lib/postgresql/data ... ok
db_1  | creating subdirectories ... ok
db_1  | selecting default max_connections ... 100
db_1  | selecting default shared_buffers ... 128MB
db_1  | selecting dynamic shared memory implementation ... posix
```

24.16.7 docker-compose exec -u postgres db psql

```
psql (10.1)
Type "help" for help.
```

```
postgres=# help
```

```
You are using psql, the command-line interface to PostgreSQL.
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit
```

```
postgres=# \h
```

Available help:		
ABORT	ALTER TRIGGER	CREATE RULE
↪ DROP GROUP	LISTEN	↪
↪ ALTER AGGREGATE	ALTER TYPE	CREATE SCHEMA
↪ DROP INDEX	LOAD	↪
↪ ALTER COLLATION	ALTER USER	CREATE SEQUENCE
↪ DROP LANGUAGE	LOCK	↪
↪ ALTER CONVERSION	ALTER USER MAPPING	CREATE SERVER
↪ DROP MATERIALIZED	MOVE	↪
↪ ALTER DATABASE	ALTER VIEW	CREATE STATISTICS
↪ DROP OPERATOR	NOTIFY	↪
↪ ALTER DEFAULT PRIVILEGES	ANALYZE	CREATE
↪ SUBSCRIPTION	DROP OPERATOR CLASS	↪
		PREPARE

(continues on next page)

(continued from previous page)

ALTER DOMAIN	BEGIN	CREATE TABLE
→ DROP OPERATOR FAMILY	PREPARE TRANSACTION	→
ALTER EVENT TRIGGER	CHECKPOINT	CREATE TABLE AS
→ DROP OWNED	REASSIGN OWNED	→
ALTER EXTENSION	CLOSE	CREATE TABLESPACE
→ DROP POLICY	REFRESH MATERIALIZED VIEW	CREATE TEXT
ALTER FOREIGN DATA WRAPPER	CLUSTER	→
→ SEARCH CONFIGURATION	DROP PUBLICATION	REINDEX
ALTER FOREIGN TABLE	COMMENT	CREATE TEXT
→ SEARCH DICTIONARY	DROP ROLE	RELEASE SAVEPOINT
ALTER FUNCTION	COMMIT	CREATE TEXT
→ SEARCH PARSER	DROP RULE	RESET
ALTER GROUP	COMMIT PREPARED	CREATE TEXT
→ SEARCH TEMPLATE	DROP SCHEMA	REVOKE
ALTER INDEX	COPY	CREATE TRANSFORM
→ DROP SEQUENCE	ROLLBACK	→
ALTER LANGUAGE	CREATE ACCESS METHOD	CREATE TRIGGER
→ DROP SERVER	ROLLBACK PREPARED	→
ALTER LARGE OBJECT	CREATE AGGREGATE	CREATE TYPE
→ DROP STATISTICS	ROLLBACK TO SAVEPOINT	→
ALTER MATERIALIZED VIEW	CREATE CAST	CREATE USER
→ DROP SUBSCRIPTION	SAVEPOINT	→
ALTER OPERATOR	CREATE COLLATION	CREATE USER
→ MAPPING	DROP TABLE	SECURITY LABEL
ALTER OPERATOR CLASS	CREATE CONVERSION	CREATE VIEW
→ DROP TABLESPACE	SELECT	→
ALTER OPERATOR FAMILY	CREATE DATABASE	DEALLOCATE
→ DROP TEXT SEARCH CONFIGURATION	SELECT INTO	→
ALTER POLICY	CREATE DOMAIN	DECLARE
→ DROP TEXT SEARCH DICTIONARY	SET	→
ALTER PUBLICATION	CREATE EVENT TRIGGER	DELETE
→ DROP TEXT SEARCH PARSER	SET CONSTRAINTS	→
ALTER ROLE	CREATE EXTENSION	DISCARD
→ DROP TEXT SEARCH TEMPLATE	SET ROLE	→
ALTER RULE	CREATE FOREIGN DATA WRAPPER	DO
→ DROP TRANSFORM	SET SESSION AUTHORIZATION	→
ALTER SCHEMA	CREATE FOREIGN TABLE	DROP ACCESS
→ METHOD	DROP TRIGGER	SET TRANSACTION
ALTER SEQUENCE	CREATE FUNCTION	DROP AGGREGATE
→ DROP TYPE	SHOW	→
ALTER SERVER	CREATE GROUP	DROP CAST
→ DROP USER	START TRANSACTION	→
ALTER STATISTICS	CREATE INDEX	DROP COLLATION
→ DROP USER MAPPING	TABLE	→
ALTER SUBSCRIPTION	CREATE LANGUAGE	DROP CONVERSION
→ DROP VIEW	TRUNCATE	→
ALTER SYSTEM	CREATE MATERIALIZED VIEW	DROP DATABASE
→ END	UNLISTEN	→
ALTER TABLE	CREATE OPERATOR	DROP DOMAIN
→ EXECUTE	UPDATE	→
ALTER TABLESPACE	CREATE OPERATOR CLASS	DROP EVENT
→ TRIGGER	EXPLAIN	VACUUM
ALTER TEXT SEARCH CONFIGURATION	CREATE OPERATOR FAMILY	DROP EXTENSION
→ FETCH	VALUES	→
ALTER TEXT SEARCH DICTIONARY	CREATE POLICY	DROP FOREIGN DATA
→ WRAPPER	GRANT	WITH
ALTER TEXT SEARCH PARSER	CREATE PUBLICATION	DROP FOREIGN
→ TABLE	IMPORT FOREIGN SCHEMA	(continues on next page)

(continued from previous page)

ALTER TEXT SEARCH TEMPLATE ↳ INSERT	CREATE ROLE	DROP FUNCTION ↳
---	-------------	--------------------

24.16.8 docker ps

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>  
↳docker ps
```

CONTAINER ID ↳STATUS d205b9239366 ↳ Up 6 minutes	IMAGE PORTS postgres:10 5432/tcp	COMMAND "docker-entrypoint.s..." postgreSQL_db_1	CREATED 6 minutes ago ↳
---	---	--	-------------------------------

24.16.9 docker exec -it d205b9239366 bash

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql>  
↳docker exec -it d205b9239366 bash
```

```
root@d205b9239366:/# ps -ef
```

UID	PID	PPID	C	S TIME	TTY	TIME	CMD
postgres	1	0	0	12:23	?	00:00:00	postgres
postgres	56	1	0	12:23	?	00:00:00	postgres: checkpointer process
postgres	57	1	0	12:23	?	00:00:00	postgres: writer process
postgres	58	1	0	12:23	?	00:00:00	postgres: wal writer process
postgres	59	1	0	12:23	?	00:00:00	postgres: autovacuum launcher ↳process
postgres	60	1	0	12:23	?	00:00:00	postgres: stats collector process
postgres	61	1	0	12:23	?	00:00:00	postgres: bgworker: logical ↳replication launcher
postgres	66	0	0	12:28	pts/0	00:00:00	/usr/lib/postgresql/10/bin/psql
postgres	78	1	0	12:28	?	00:00:00	postgres: postgres postgres [local] ↳idle
root	110	0	0	12:45	pts/1	00:00:00	bash
root	114	110	0	12:45	pts/1	00:00:00	ps -ef

```
root@d205b9239366:/# uname -a
```

```
Linux d205b9239366 4.9.60-linuxkit-aufs #1 SMP Mon Nov 6 16:00:12 UTC 2017 x86_64 GNU/  
↳Linux
```

```
root@d205b9239366:/# which psql
```

```
/usr/bin/psql
```

24.16.10 Mardi 30 janvier 2018

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
02b2487f304e	postgres:10.1	"docker-entrypoint.s..."	18 seconds ago
↑ Up 16 seconds	5432/tcp	postgres_test	

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker exec -it 02b2487f304e bash
```

```
root@02b2487f304e:/# psql -U postgres
psql (10.1)
Type "help" for help.

postgres=# \dt
Did not find any relations.
postgres=# \l
List of databases
   Name    | Owner | Encoding | Collate | Ctype | Access
privileges
----+-----+-----+-----+-----+-----+
db_id3_intranet | id3admin | UTF8      | en_US.utf8 | en_US.utf8 |
postgres        | postgres  | UTF8      | en_US.utf8 | en_US.utf8 |
template0       | postgres  | UTF8      | en_US.utf8 | en_US.utf8 | =c/postgres
+
template1       | postgres  | UTF8      | en_US.utf8 | en_US.utf8 | =c/postgres
+
postgres=CTc/postgres
(4 rows)
```

24.16.10.1 docker-compose.yml

```
version: "3"

services:
  db:
    image: postgres:10.1
    container_name: container_intranet
    volumes:
      - volume_intranet:/var/lib/postgresql/data/

volumes:
  volume_intranet:
```

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
47501acda106	postgres:10.1	"docker-entrypoint.s..."	15 minutes ago
Up 15 minutes	5432/tcp	container_intranet	

24.16.10.2 docker volume ls

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker volume ls
```

DRIVER	VOLUME NAME
local	postgresql_volume_intranet

24.16.10.3 docker volume inspect postgresql_volume_intranet

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker volume inspect postgresql_volume_intranet
```

```
[{"Driver": "local", "Name": "postgresql_volume_intranet", "Mountpoint": "/var/lib/docker/volumes/postgresql_volume_intranet/_data", "Options": {}, "Scope": "local", "Labels": {"com.docker.compose.project": "postgresql", "com.docker.compose.volume": "volume_intranet"}, "CreatedAt": "2018-01-30T12:14:30Z"}]
```

24.16.10.4 docker exec -it 47501acda106 bash

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker exec -it 47501acda106 bash
```

24.16.10.5 psql -U postgres

```
root@47501acda106:/# psql -U postgres
```

```
psql (10.1)
Type "help" for help.
```

24.16.10.6 I (liste des bases de données)

```
postgres=# \l
```

List of databases						
Name	Owner	Encoding	Collate	Ctype	Access privileges	
postgres	postgres	UTF8	en_US.utf8	en_US.utf8		
template0	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres	+
						postgres=CTc/
→postgres						
template1	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres	+
						postgres=CTc/
→postgres						
(3 rows)						

24.16.10.7 CREATE USER id3admin WITH PASSWORD 'id338';

```
postgres=# CREATE USER id3admin WITH PASSWORD 'id338';
```

```
CREATE ROLE
```

24.16.10.8 CREATE DATABASE db_id3_intranet WITH OWNER = id3admin ENCODING = 'UTF8' CONNECTION LIMIT = -1;

```
postgres=# CREATE DATABASE db_id3_intranet WITH OWNER = id3admin ENCODING = 'UTF8' →
→CONNECTION LIMIT = -1;
```

```
CREATE DATABASE
```

24.16.10.9 I

```
postgres=# \l
```

List of databases						
Name	Owner	Encoding	Collate	Ctype	Access privileges	
→privileges						
db_id3_intranet	id3admin	UTF8	en_US.utf8	en_US.utf8		
postgres	postgres	UTF8	en_US.utf8	en_US.utf8		
template0	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres	+
→ +						
→postgres=CTc/postgres						
template1	postgres	UTF8	en_US.utf8	en_US.utf8	=c/postgres	+
→ +						
→postgres=CTc/postgres						
(4 rows)						

24.16.10.10 docker-compose run db env

See also:

- <https://realpython.com/blog/python/django-development-with-docker-compose-and-machine/>

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker-compose run db env
```

```
LANG=en_US.utf8
HOSTNAME=7dc6fce71c87
PG_MAJOR=10
PWD=/
HOME=/root
PG_VERSION=10.1-1.pgdg90+1
GOSU_VERSION=1.10
PGDATA=/var/lib/postgresql/data
TERM=xterm
SHLVL=0
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/usr/lib/postgresql/
→10/bin
```

24.16.10.11 docker-compose config

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker-compose config
```

```
services:
  db:
    container_name: container_intranet
    environment:
      LANG: fr_FR.utf8
    image: postgres:10.1
    ports:
      - 5432:5432/tcp
    volumes:
      - volume_intranet:/var/lib/postgresql/data/:rw
      - Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql:/code:rw
  version: '3.0'
  volumes:
    volume_intranet: {}
```

24.16.11 Import de la base de données

```
pg_restore --dbname=db_id3_intranet --username=id3admin -f db_id3_intranet.sql
```

24.16.12 Mercredi 31 janvier 2018 : export/import d'une base de données PostgreSQL (tutoriel PostgreSQL)

24.16.12.1 pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet

```
pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet
```

```
PS C:\Tmp> pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet
Mot de passe :
PS C:\Tmp> _
```

Fig. 14: pg_dump -U postgres --clean --create -f db.dump.sql db_id3_intranet

24.16.12.2 Entête de db.dump

C'est du format texte.

```
--  
-- PostgreSQL database dump  
--  
  
-- Dumped from database version 10.1  
-- Dumped by pg_dump version 10.1  
  
-- Started on 2018-01-31 10:16:48  
  
SET statement_timeout = 0;  
SET lock_timeout = 0;  
SET idle_in_transaction_session_timeout = 0;  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = on;  
SET check_function_bodies = false;  
SET client_min_messages = warning;  
SET row_security = off;  
  
DROP DATABASE db_id3_intranet;  
--  
-- TOC entry 3644 (class 1262 OID 16394)  
-- Name: db_id3_intranet; Type: DATABASE; Schema: -; Owner: id3admin  
--  
  
CREATE DATABASE db_id3_intranet WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_  
→COLLATE = 'French_France.1252' LC_CTYPE = 'French_France.1252';  
  
ALTER DATABASE db_id3_intranet OWNER TO id3admin;  
  
\connect db_id3_intranet  
  
SET statement_timeout = 0;  
SET lock_timeout = 0;  
SET idle_in_transaction_session_timeout = 0;  
SET client_encoding = 'UTF8';  
SET standard_conforming_strings = on;
```

(continues on next page)

(continued from previous page)

```
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;
```

24.16.12.3 Expérience substitution de db_id3_save à db_id3_intranet

On substitue db_id3_save à db_id3_intranet. On espère donc créer une copie de la base de données db_id3_intranet. Comme le fichier est au format texte, on peut utiliser psql pour l'import.

```
-- 
-- PostgreSQL database dump
-- 

-- Dumped from database version 10.1
-- Dumped by pg_dump version 10.1

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

-- 
-- Name: db_id3_save; Type: DATABASE; Schema: -; Owner: id3admin
-- 

CREATE DATABASE db_id3_save WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE =
←'French_France.1252' LC_CTYPE = 'French_France.1252';

ALTER DATABASE db_id3_save OWNER TO id3admin;

\connect db_id3_save

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

-- 
-- Name: db_id3_save; Type: COMMENT; Schema: -; Owner: id3admin
-- 

COMMENT ON DATABASE db_id3_save IS 'La base db_id3_save';
```

24.16.12.4 psql -U postgres -f .\db.dump.sql

```
psql -U postgres -f .\db.dump.sql
```

```
ALTER TABLE
GRANT
```

OK, tout s'est bien passé.

```
PS C:\Tmp> psql -U postgres
Mot de passe pour l'utilisateur postgres :
pgsql (10.1)
Attention : l'encodage console (850) diffère de l'encodage Windows (1252).
Les caractères 8 bits peuvent ne pas fonctionner correctement.
Voir la section « Notes aux utilisateurs de Windows » de la page
référence de psql pour les détails.
Saisissez « help » pour l'aide.

postgres=# \l
           Liste des bases de données
   Nom   | Propriétaire | Encodage | Collationnement | Type caract. | Droits d'accès
---+-----+-----+-----+-----+-----+
db_id3_intranet | id3admin | UTF8 | French_France.1252 | French_France.1252 |
db_id3_save     | id3admin | UTF8 | French_France.1252 | French_France.1252 |
db_test         | id3admin | UTF8 | French_France.1252 | French_France.1252 |
postgres        | postgres  | UTF8 | French_France.1252 | French_France.1252 |
template0       | postgres  | UTF8 | French_France.1252 | French_France.1252 |
template1       | postgres  | UTF8 | French_France.1252 | French_France.1252 |
(6 lignes)
```

Fig. 15: psql -U postgres -f .\db.dump.sql

On voit aussi que l'encodage French_France.1252 va peut-être poser des problèmes dans l'image Docker actuelle.

```
postgres=# \l
           Liste des bases de données
   Nom   | Propriétaire | Encodage | Collationnement | Type caract. |
   | Droits d'accès
---+-----+-----+-----+-----+
db_id3_intranet | id3admin | UTF8 | French_France.1252 | French_France.1252 |
db_id3_save     | id3admin | UTF8 | French_France.1252 | French_France.1252 |
db_test         | id3admin | UTF8 | French_France.1252 | French_France.1252 |
postgres        | postgres  | UTF8 | French_France.1252 | French_France.1252 |
template0       | postgres  | UTF8 | French_France.1252 | French_France.1252 |
| =c/postgres    +
|                   |           |           |           |           |
|                   | postgres=CTc/postgres
| template1       | postgres  | UTF8 | French_France.1252 | French_France.1252 |
| =c/postgres    +
|                   |           |           |           |           |
|                   | postgres=CTc/postgres
(6 lignes)
```

Sur Docker on a:

```
root@02b2487f304e:/# psql -U postgres
psql (10.1)
Type "help" for help.

postgres=# \dt
Did not find any relations.
postgres=# \l
                                         List of databases
   Name    | Owner | Encoding | Collate | Ctype | Access
postgres  | postgres | UTF8    | en_US.utf8 | en_US.utf8 | c/postgres
template0 | postgres | UTF8    | en_US.utf8 | en_US.utf8 | =c/postgres
template1 | postgres | UTF8    | en_US.utf8 | en_US.utf8 | =c/postgres
postgres=CTc/postgres
(4 rows)
```

On suit les conseils donnés *ici*: On essaye déjà avec la langue allemande et on essayera avec *French_France.1252* ?

Dockerfile:

```
FROM postgres:10.1
RUN localedef -i de_DE -c -f UTF-8 -A /usr/share/locale/locale.alias de_DE.UTF-8
ENV LANG de_DE.utf8
```

```
C:\WINDOWS\system32\windowspowershell\v1.0\powershell.exe
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql> docker-compose stop
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql> docker-compose rm
Going to remove container_intranet
Are you sure? [Y/N] y
Removing container_intranet ... done
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker\tutoriels\postgresql> -
```

24.16.12.5 docker-compose stop

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker-compose stop
```

```
Stopping container_intranet ... done
```

24.16.12.6 docker-compose build

```
PS Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\tutoriels\postgresql> docker-compose build
```

```
Building db
Step 1/3 : FROM postgres:10.1
--> ec61d13c8566
Step 2/3 : RUN localedef -i de_DE -c -f UTF-8 -A /usr/share/locale/locale.alias de_DE.
--> UTF-8
--> Running in 19e95836a1ce
Removing intermediate container 19e95836a1ce
--> 331ee9213868
Step 3/3 : ENV LANG de_DE.utf8
--> Running in 852054da9e27
Removing intermediate container 852054da9e27
--> 56dd534c98f7
Successfully built 56dd534c98f7
Successfully tagged postgres:10.1
```

24.16.13 CREATE DATABASE db_id3_save WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE = 'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';

```
postgres=# CREATE DATABASE db_id3_save WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_
-->COLLATE = 'fr_FR.UTF-8' LC_CTYPE = 'fr_FR.UTF-8';
```

```
CREATE DATABASE
```

24.17 Docker OpenLDAP

See also:

- <https://github.com/osixia/docker-openldap>

Contents

- *Docker OpenLDAP*

CHAPTER 25

Exemples Docker labs

See also:

- <https://docs.docker.com/samples/#tutorial-labs>

25.1 Samples Docker labs

25.1.1 Samples Docker labs beginner

See also:

- <https://github.com/docker/labs/tree/master/beginner/>
- https://hub.docker.com/_/hello-world/
- <https://raw.githubusercontent.com/docker-library/hello-world/master/hello.c>

Contents

- *Samples Docker labs beginner*
 - *Setup*
 - *docker run hello-world*
 - * *hello.c*
 - * *Dockerfile.build*
 - *Running your first container : docker pull alpine*
 - * *docker pull alpine*
 - * *docker images*
 - * *docker run alpine ls -l*

```
* docker ps -a
* docker run -it alpine /bin/sh
- docker run --help
- docker inspect alpine
- Next Steps: 2.0 Webapps with Docker
```

25.1.1.1 Setup

See also:

- <https://github.com/docker/labs/blob/master/beginner/chapters/setup.md>

25.1.1.2 docker run hello-world

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker>docker run hello-
world
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

```
https://cloud.docker.com/
```

For more examples and ideas, visit:

```
https://docs.docker.com/engine/userguide/
```

25.1.1.2.1 hello.c

See also:

- <https://github.com/docker-library/hello-world/blob/master/hello.c>

```
1 // #include <unistd.h>
2 #include <sys/syscall.h>
3
4 #ifndef DOCKER_IMAGE
5     #define DOCKER_IMAGE "hello-world"
6 #endif
```

(continues on next page)

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [version 10.0.15063]
(c) 2017 Microsoft Corporation. Tous droits réservés.

Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker>docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
 executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
 to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
```

(continued from previous page)

```
7      #ifndef DOCKER_GREETING
8          #define DOCKER_GREETING "Hello from Docker!"
9      #endif
10
11
12     #ifndef DOCKER_ARCH
13         #define DOCKER_ARCH "amd64"
14     #endif
15
16     const char message[] =
17         "\n"
18         DOCKER_GREETING "\n"
19         "This message shows that your installation appears to be working_\n"
20         "correctly.\n"
21         "\n"
22         "To generate this message, Docker took the following steps:\n"
23         " 1. The Docker client contacted the Docker daemon.\n"
24         " 2. The Docker daemon pulled the \"\" DOCKER_IMAGE \"\" image from the_\n"
25         "Docker Hub.\n"
26         "    (" DOCKER_ARCH ")\n"
27         " 3. The Docker daemon created a new container from that image which_\n"
28         "runs the\n"
29         "    executable that produces the output you are currently reading.\n"
30         " 4. The Docker daemon streamed that output to the Docker client, which_\n"
31         "sent it\n"
32         "    to your terminal.\n"
33         "\n"
34         "To try something more ambitious, you can run an Ubuntu container with:\n"
35         "\n"
36         "$ docker run -it ubuntu bash\n"
37         "\n"
38         "Share images, automate workflows, and more with a free Docker ID:\n"
39         " https://cloud.docker.com/\n"
40         "\n"
```

(continues on next page)

(continued from previous page)

```

36     "For more examples and ideas, visit:\n"
37     " https://docs.docker.com/engine/userguide/\n"
38     "\n";
39
40     void _start() {
41         //write(1, message, sizeof(message) - 1);
42         syscall(SYS_write, 1, message, sizeof(message) - 1);
43
44         //__exit(0);
45         syscall(SYS_exit, 0);
46     }

```

25.1.1.2.2 Dockerfile.build

```

# explicitly use Debian for maximum cross-architecture compatibility
FROM debian:stretch-slim

RUN dpkg --add-architecture i386

RUN apt-get update && apt-get install -y --no-install-recommends \
    gcc \
    libc6-dev \
    make \
    \
    libc6-dev:i386 \
    libgcc-6-dev:i386 \
    \
    libc6-dev-arm64-cross \
    libc6-dev-armel-cross \
    libc6-dev-armhf-cross \
    libc6-dev-ppc64el-cross \
    libc6-dev-s390x-cross \
    \
    gcc-aarch64-linux-gnu \
    gcc-arm-linux-gnueabi \
    gcc-arm-linux-gnueabihf \
    gcc-powerpc64le-linux-gnu \
    gcc-s390x-linux-gnu \
    \
    file \
&& rm -rf /var/lib/apt/lists/*

WORKDIR /usr/src/hello
COPY . .

RUN set -ex; \
    make clean all test \
        TARGET_ARCH='amd64' \
        CC='x86_64-linux-gnu-gcc' \
        STRIP='x86_64-linux-gnu-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='arm32v5' \

```

(continues on next page)

(continued from previous page)

```

CC='arm-linux-gnueabi-gcc' \
STRIP='arm-linux-gnueabi-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='arm32v7' \
        CC='arm-linux-gnueabihf-gcc' \
        STRIP='arm-linux-gnueabihf-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='arm64v8' \
        CC='aarch64-linux-gnu-gcc' \
        STRIP='aarch64-linux-gnu-strip'

RUN set -ex; \
    make clean all test \
        TARGET_ARCH='i386' \
        CC='gcc -m32 -L/usr/lib/gcc/i686-linux-gnu/6' \
        STRIP='x86_64-linux-gnu-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='ppc64le' \
        CC='powerpc64le-linux-gnu-gcc' \
        STRIP='powerpc64le-linux-gnu-strip'

RUN set -ex; \
    make clean all \
        TARGET_ARCH='s390x' \
        CC='s390x-linux-gnu-gcc' \
        STRIP='s390x-linux-gnu-strip'

RUN find \(( -name 'hello' -or -name 'hello.txt' \) ) -exec file '{}' + -exec ls -lh '{}' \
←' + 

CMD ["/./amd64/hello-world/hello"]

```

25.1.1.3 Running your first container : docker pull alpine

See also:

- <https://github.com/docker/labs/blob/master/beginner/chapters/alpine.md>
- *Images Alpine*

25.1.1.3.1 docker pull alpine

```
docker pull alpine
```

25.1.1.3.2 docker images

REPOSITORY	TAG	IMAGE ID	CREATED	
id3pvergain/get-started	part2	ed5b70620e49	25 hours ago	[link]
friendlyhello	latest	ed5b70620e49	25 hours ago	[link]
alpine	latest	3fd9065eaf02	6 days ago	[link]
wordpress	latest	28084cde273b	7 days ago	[link]
centos	latest	ff426288ea90	7 days ago	[link]
nginx	latest	3f8a4339aadd	2 weeks ago	[link]
ubuntu	latest	00fd29ccc6f1	4 weeks ago	[link]
python	2.7-slim	4fd30fc83117	5 weeks ago	[link]
hello-world	latest	f2a91732366c	8 weeks ago	[link]
docker4w/nsenter-dockerd	latest	cae870735e91	2 months ago	[link]

25.1.1.3.3 docker run alpine ls -l

Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_docker>docker run alpine ls	[link]
<code>→-l</code>	
<code>total 52</code>	
<code>drwxr-xr-x 2 root root 4096 Jan 9 19:37 bin</code>	
<code>drwxr-xr-x 5 root root 340 Jan 16 08:57 dev</code>	
<code>drwxr-xr-x 1 root root 4096 Jan 16 08:57 etc</code>	
<code>drwxr-xr-x 2 root root 4096 Jan 9 19:37 home</code>	
<code>drwxr-xr-x 5 root root 4096 Jan 9 19:37 lib</code>	
<code>drwxr-xr-x 5 root root 4096 Jan 9 19:37 media</code>	
<code>drwxr-xr-x 2 root root 4096 Jan 9 19:37 mnt</code>	
<code>dr-xr-xr-x 127 root root 0 Jan 16 08:57 proc</code>	
<code>drwx----- 2 root root 4096 Jan 9 19:37 root</code>	
<code>drwxr-xr-x 2 root root 4096 Jan 9 19:37 run</code>	
<code>drwxr-xr-x 2 root root 4096 Jan 9 19:37 sbin</code>	
<code>drwxr-xr-x 2 root root 4096 Jan 9 19:37 srv</code>	
<code>dr-xr-xr-x 13 root root 0 Jan 15 15:33 sys</code>	
<code>drwxrwxrwt 2 root root 4096 Jan 9 19:37 tmp</code>	
<code>drwxr-xr-x 7 root root 4096 Jan 9 19:37 usr</code>	
<code>drwxr-xr-x 11 root root 4096 Jan 9 19:37 var</code>	

What happened? Behind the scenes, a lot of stuff happened. When you call run:

- The Docker client contacts the Docker daemon
- The Docker daemon checks local store if the image (alpine in this case) is available locally, and if not, downloads it from Docker Store. (Since we have issued docker pull alpine before, the download step is not necessary)

- The Docker daemon creates the container and then runs a command in that container.
- The Docker daemon streams the output of the command to the Docker client

When you run docker run alpine, you provided a command (ls -l), so Docker started the command specified and you saw the listing.

25.1.1.3.4 docker ps -a

Liste des conteneurs qui ont tourné à un moment donné.

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS			NAMES
cb62ace67ba4	alpine	"ls -l"	20 hours ago
→ minutes ago	Exited (0) 20 minutes ago		eager_heisenberg
685915373a4c	hello-world	"/hello"	2 hours ago
→ ago	Exited (0) 2 hours ago		gallant_wright
e150d0531321	alpine	"/bin/sh"	18 hours ago
→ ago	Exited (0) 18 hours ago		objective_curran
7d6e93a39de5	alpine	"/bin/sh"	18 hours ago
→ ago	Exited (0) 18 hours ago		amazing_knuth
807d38ada261	ubuntu	"/bin/bash"	18 hours ago
→ ago	Exited (127) 18 hours ago		confident_bassi
eebf7e801b96	ubuntu	"/bin/bash"	18 hours ago
→ ago	Exited (0) 13 minutes ago		wonderful_blackwell
c31e71b41bdb	id3pvergain/get-started:part2	"python app.py"	22 hours ago
→ ago	Exited (137) 20 hours ago		getstartedlab_web.3.
→ kv05oigiytufm5wsuvnp4guoj			
8780b68999cf	id3pvergain/get-started:part2	"python app.py"	22 hours ago
→ ago	Exited (137) 20 hours ago		getstartedlab_web.4.
→ as0f73cwm518fibwnjd60yfyw			
f45453da50cf	id3pvergain/get-started:part2	"python app.py"	23 hours ago
→ ago	Exited (137) 20 hours ago		youthful_wilson
b47fd081642e	id3pvergain/get-started:part2	"python app.py"	23 hours ago
→ ago	Exited (137) 20 hours ago		admiring_lumiere
06193b763075	friendlyhello	"python app.py"	24 hours ago
→ ago	Exited (137) 23 hours ago		boring_goodall
16eca9f1274e	friendlyhello	"python app.py"	26 hours ago
→ ago	Exited (255) 24 hours ago	0.0.0.0:4000->80/tcp	stoic_lalande
fb92255412cf	hello-world	"hello"	3 days ago
→ ago	Exited (0) 3 days ago		infallible_kepler
dd8ca306fb5b	hello-world	"hello"	4 days ago
→ ago	Exited (0) 4 days ago		musing_hopper
4d1e5f24ba8e	nginx	"nginx -g 'daemon off...'"	4 days ago
→ ago	Exited (0) 4 days ago		webserver

25.1.1.3.5 docker run -it alpine /bin/sh

```
C:\Tmp>docker run -it alpine /bin/sh
```

```
/ # uname -a
```

```
Linux 2b8ffff5f4068 4.9.60-linuxkit-aufs #1 SMP Mon Nov 6 16:00:12 UTC 2017 x86_64
↳Linux
```

```
/ # ls
```

```
bin dev etc home lib media mnt proc root run sbin srv
↳sys tmp usr var
```

Running the run command with the -it flags attaches us to an interactive tty in the container. Now you can run as many commands in the container as you want. Take some time to run your favorite commands.

That concludes a whirlwind tour of the docker run command which would most likely be the command you'll use most often.

It makes sense to spend some time getting comfortable with it.

To find out more about run, use docker run –help to see a list of all flags it supports.

As you proceed further, we'll see a few more variants of docker run.

25.1.1.4 docker run –help

```
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Run a command **in** a new container

Options:

--add-host <i>list</i>	Add a custom host-to-IP mapping (host:ip)
-a, --attach <i>list</i>	Attach to STDIN, STDOUT or STDERR
--blkio-weight <i>uint16</i>	Block IO (relative weight), between 10 and 1000, or 0 to disable
--(default 0)	--blkio-weight-device <i>list</i> Block IO weight (relative device weight)
--(default [])	--cap-add <i>list</i> Add Linux capabilities --cap-drop <i>list</i> Drop Linux capabilities --cgroup-parent <i>string</i> Optional parent cgroup for the container
--cidfile <i>string</i>	Write the container ID to the file
--cpu-period <i>int</i>	Limit CPU CFS (Completely Fair Scheduler)
--cpu-quota <i>int</i>	Limit CPU CFS (Completely Fair Scheduler)
--quota	--cpu-rt-period <i>int</i> Limit CPU real-time period in microseconds
--cpu-rt-runtime <i>int</i>	Limit CPU real-time runtime in microseconds
-c, --cpu-shares <i>int</i>	CPU shares (relative weight)

(continues on next page)

(continued from previous page)

--cpus decimal	Number of CPUs
--cpuset-cpus string	CPUs in which to allow execution <i>(0..3, 0..1)</i>
--cpuset-mems string	MEMs in which to allow execution <i>(0..3, 0..1)</i>
-d, --detach	Run container in background and print █
↳ container ID	Override the key sequence for detaching █
↳ a container	Add a host device to the container Add a rule to the cgroup allowed devices █
↳ list	Limit read rate (bytes per second) from a █
↳ device (default [])	Limit read rate (IO per second) from a █
↳ device (default [])	Limit write rate (bytes per second) to a █
↳ a device (default [])	Limit write rate (IO per second) to a █
↳ device (default [])	Skip image verification (default true) Set custom DNS servers Set DNS options Set custom DNS search domains Overwrite the default ENTRYPOINT of the █
↳ image	Set environment variables
-e, --env list	Read in a file of environment variables Expose a port or a range of ports Add additional groups to join Command to run to check health Time between running the check <i>(ms s m h) █</i>
↳ (default 0s)	Consecutive failures needed to report █
↳ unhealthy	Start period for the container to initialize █
↳ before starting	health- <i>(ms s m h) █</i>
↳ retries countdown	Maximum time to allow one check to run █
↳ (default 0s)	Print usage
-h, --hostname string	Container host name

(continues on next page)

(continued from previous page)

--init	Run an init inside the container that processes
→forwards signals and reaps	
-i, --interactive	Keep STDIN open even if not attached
--ip string	IPv4 address (e.g., 172.30.100.104)
--ip6 string	IPv6 address (e.g., 2001:db8::33)
--ipc string	IPC mode to use
--isolation string	Container isolation technology
--kernel-memory bytes	Kernel memory limit
-l, --label list	Set meta data on a container
--label-file list	Read in a line delimited file of labels
--link list	Add link to another container
--link-local-ip list	Container IPv4/IPv6 link-local addresses
--log-driver string	Logging driver for the container
--log-opt list	Log driver options
--mac-address string	Container MAC address (e.g.,
→92:d0:c6:0a:29:33)	
-m, --memory bytes	Memory limit
--memory-reservation bytes	Memory soft limit
--memory-swap bytes	Swap limit equal to memory plus swap: '-1'
→to enable unlimited swap	
--memory-swappiness int	Tune container memory swappiness (0 to 100)
→(default -1)	
--mount mount	Attach a filesystem mount to the container
--name string	Assign a name to the container
--network string	Connect a container to a network (default
→"default")	
--network-alias list	Add network-scoped alias for the container
--no-healthcheck	Disable any container-specified HEALTHCHECK
--oom-kill-disable	Disable OOM Killer
--oom-score-adj int	Tune host's OOM preferences (-1000 to 1000)
--pid string	PID namespace to use
--pids-limit int	Tune container pids limit (set -1 for
→unlimited)	
--platform string	Set platform if server is multi-
→platform capable	
--privileged	Give extended privileges to this container
-p, --publish list	Publish a container's port(s) to the host
-P, --publish-all	Publish all exposed ports to random
→ports	
--read-only	Mount the container's root filesystem
→as read only	

(continues on next page)

(continued from previous page)

--restart string	Restart policy to apply when a container
→ exits (default "no")	Automatically remove the container when it
→ exits	Run time to use for this container
--runtime string	Security Options
--security-opt list	Size of /dev/shm
--shm-size bytes	Proxy received signals to the process
--sig-proxy	
→ (default true)	Signal to stop a container
--stop-signal string	(default
→ "15")	Timeout (in seconds) to stop a container
--stop-timeout int	
--storage-opt list	Storage driver options for the container
--sysctl map	Sysctl options (default map[{}])
--tmpfs list	Mount a tmpfs directory
-t, --tty	Allocate a pseudo-TTY
--ulimit ulimit	Ulimit options (default [])
-u, --user string	Username or UID (format: <name uid>)
→ [:<group gid>]	
--userns string	User namespace to use
--uts string	UTS namespace to use
-v, --volume list	Bind mount a volume
--volume-driver string	Optional volume driver for the container
--volumes-from list	Mount volumes from the specified
→ container(s)	
-w, --workdir string	Working directory inside the container

25.1.1.5 docker inspect alpine

C:\Tmp>docker inspect alpine

```
[
  {
    "Id": "sha256:3fd9065eaf02feaf94d68376da52541925650b81698c53c6824d92ff63f98353",
    "RepoTags": [
      "alpine:latest"
    ],
    "RepoDigests": [
      "alpine@sha256:7df6db5aa61ae9480f52f0b3a06a140ab98d427f86d8d5de0bedab9b8df6b1c0"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2018-01-09T21:10:58.579708634Z",
  }
]
```

(continues on next page)

(continued from previous page)

```

    "Container": [
        {"Id": "30e1a2427aa2325727a092488d304505780501585a6ccf5a6a53c4d83a826101",
         "ContainerConfig": {
             "Hostname": "30e1a2427aa2",
             "Domainname": "",
             "User": "",
             "AttachStdin": false,
             "AttachStdout": false,
             "AttachStderr": false,
             "Tty": false,
             "OpenStdin": false,
             "StdinOnce": false,
             "Env": [
                 "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin"
             ],
             "Cmd": [
                 "/bin/sh",
                 "-c",
                 "#(nop) ",
                 "CMD [\"/bin/sh\"]"
             ],
             "ArgsEscaped": true,
             "Image": "sha256:fbef17698ac8605733924d5662f0cbfc0b27a51e83ab7d7a4b8d8a9a9fe0d1c2",
             "Volumes": null,
             "WorkingDir": "",
             "Entrypoint": null,
             "OnBuild": null,
             "Labels": {}
         },
         "DockerVersion": "17.06.2-ce",
         "Author": "",
         "Config": {
             "Hostname": "",
             "Domainname": "",
             "User": "",
             "AttachStdin": false,
             "AttachStdout": false,
             "AttachStderr": false,
             "Tty": false,
             "OpenStdin": false,
             "StdinOnce": false,
             "Env": [
                 "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin"
             ],
             "Cmd": [
                 "/bin/sh"
             ],
             "ArgsEscaped": true,
             "Image": "sha256:fbef17698ac8605733924d5662f0cbfc0b27a51e83ab7d7a4b8d8a9a9fe0d1c2",
             "Volumes": null,
             "WorkingDir": "",
             "Entrypoint": null,
             "OnBuild": null,
         }
     ]
}

```

(continues on next page)

(continued from previous page)

```

        "Labels": null
    },
    "Architecture": "amd64",
    "Os": "linux",
    "Size": 4147781,
    "VirtualSize": 4147781,
    "GraphDriver": {
        "Data": {
            "MergedDir": "/var/lib/docker/overlay2/
→e4af82b9362c03a84a71a8449c41a37c94592f1e5c2ef1d4f43a255b0a4ee2bd/merged",
            "UpperDir": "/var/lib/docker/overlay2/
→e4af82b9362c03a84a71a8449c41a37c94592f1e5c2ef1d4f43a255b0a4ee2bd/diff",
            "WorkDir": "/var/lib/docker/overlay2/
→e4af82b9362c03a84a71a8449c41a37c94592f1e5c2ef1d4f43a255b0a4ee2bd/work"
        },
        "Name": "overlay2"
    },
    "RootFS": {
        "Type": "layers",
        "Layers": [
            "sha256:cd7100a72410606589a54b932cabd804a17f9ae5b42a1882bd56d263e02b6215"
        ]
    },
    "Metadata": {
        "LastTagTime": "0001-01-01T00:00:00Z"
    }
}
]

```

25.1.1.6 Next Steps: 2.0 Webapps with Docker

See also:

<https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md>

For the next step in the tutorial, head over to *2.0 Webapps with Docker*.

25.1.2 2) Webapps with Docker (Python + Flask)

See also:

- <https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md>
- <https://github.com/docker/labs/tree/master/beginner/static-site>
- <https://hub.docker.com/r/dockersamples/static-site/>

Contents

- 2) *Webapps with Docker (Python + Flask)*
 - *Introduction*
 - *Run a static website in a container : docker run -d dockersamples/static-site*

- *docker images*
- *docker run -name static-site -e AUTHOR="patrick.vergain" -d -P dockersamples/static-site*
- *docker port static-site*
- *docker run -name static-site-2 -e AUTHOR="patrick.vergain" -d -p 8888:80 dockersamples/static-site*
- *docker stop static-site*
- *docker rm static-site*
- *Let's use a shortcut to remove the second site: docker rm -f static-site-2*
- *Docker Images*
- *docker pull ubuntu:16.04*
- *Create your first image*
- *Create a Python Flask app that displays random cat pix*
 - * *app.py*
 - * *requirements.txt*
 - * *templates/index.html*
 - * *Write a Dockerfile*
 - *FROM alpine:3.5*
 - *RUN apk add --update py2-pip*
 - *COPY requirements.txt /usr/src/app/*
 - *COPY app.py /usr/src/app/*
 - *EXPOSE 5000*
 - *CMD ["python", "/usr/src/app/app.py"]*
 - * *Build the image (docker build -t id3pvergain/myfirstapp)*
 - * *docker images*
 - * *Run your image (docker run -p 8888:5000 -name myfirstapp id3pvergain/myfirstapp)*
 - * *Push your image (docker push id3pvergain/myfirstapp)*
 - *docker login*
 - *docker push id3pvergain/myfirstapp*
 - * *docker rm -f myfirstapp*
 - * *docker ps*
- *Dockerfile commands summary*
 - * *FROM*
 - * *RUN*
 - * *COPY*
 - * *CMD*
 - * *EXPOSE*

* **PUSH**

- *Next Steps : Deploying an app to a Swarm*

25.1.2.1 Introduction

Great! So you have now looked at docker run, played with a Docker container and also got the hang of some terminology.

Armed with all this knowledge, you are now ready to get to the real stuff, deploying web applications with Docker.

25.1.2.2 Run a static website in a container : docker run -d dockersamples/static-site

Note: Code for this section is in this repo in the static-site directory.

Let's start by taking baby-steps. First, we'll use Docker to run a static website in a container.

The website is based on an existing image.

We'll pull a Docker image from Docker Store, run the container, and see how easy it is to set up a web server.

The image that you are going to use is a single-page website that was already created for this demo and is available on the Docker Store as dockersamples/static-site.

You can download and run the image directly in one go using docker run as follows:

```
docker run -d dockersamples/static-site
```

```
C:\Tmp>docker run -d dockersamples/static-site
```

```
Unable to find image 'dockersamples/static-site:latest' locally
latest: Pulling from dockersamples/static-site
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
716f7a5f3082: Pull complete
7b10f03a0309: Pull complete
aff3ab7e9c39: Pull complete
Digest: sha256:daa686c61d7d239b7977e72157997489db49f316b9b9af3909d9f10fd28b2dec
Status: Downloaded newer image for dockersamples/static-site:latest
3bf76a82d6127dfd775f0eb6a5ed20ce275ad7eaf02b18b2ce50bd96df1432ba
```

25.1.2.3 docker images

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
ubuntu	trusty	02a63d8b2bfa	17 hours ago
id3pvergain/get-started	part2	ed5b70620e49	31 hours ago
friendlyhello	latest	ed5b70620e49	31 hours ago
			(continues on next page)

(continued from previous page)

alpine	latest	3fd9065eaf02	6 days ago	[1]
↳ 4.15MB				
wordpress	latest	28084cde273b	7 days ago	[1]
↳ 408MB				
centos	latest	ff426288ea90	7 days ago	[1]
↳ 207MB				
nginx	latest	3f8a4339aadd	2 weeks ago	[1]
↳ 108MB				
ubuntu	latest	00fd29ccc6f1	4 weeks ago	[1]
↳ 111MB				
python	2.7-slim	4fd30fc83117	5 weeks ago	[1]
↳ 138MB				
hello-world	latest	f2a91732366c	8 weeks ago	[1]
↳ 1.85kB				
docker4w/nsenter-dockerd	latest	cae870735e91	2 months ago	[1]
↳ 187kB				
dockersamples/static-site	latest	f589ccde7957	22 months ago	[1]
↳ 191MB				

25.1.2.4 docker run --name static-site -e AUTHOR="patrick.vergain" -d -P dockersamples/static-site

```
C:\Tmp>docker run --name static-site -e AUTHOR="patrick.vergain" -d -P dockersamples/
↳static-site
```

```
554e21d4b723a49e4b2019497d4411d955de2175e8b216a126d3a0c214ca9458
```

In the above command:

- -d will create a container with the process detached from our terminal
- -P will publish all the exposed container ports to random ports on the Docker host
- -e is how you pass environment variables to the container
- --name allows you to specify a container name
- AUTHOR is the environment variable name and Your Name is the value that you can pass

25.1.2.5 docker port static-site

```
docker port static-site
```

```
443/tcp -> 0.0.0.0:32768
80/tcp -> 0.0.0.0:32769
```

If you are running Docker for Mac, Docker for Windows, or Docker on Linux, you can open http://localhost:{YOUR_PORT_FOR_80/tcp}. For our example this is <http://localhost:32769/>

25.1.2.6 docker run --name static-site-2 -e AUTHOR="patrick.vergain" -d -p 8888:80 dockersamples/static-site

```
C:\Tmp>docker run --name static-site-2 -e AUTHOR="patrick.vergain" -d -p 8888:80
↳dockersamples/static-site
```

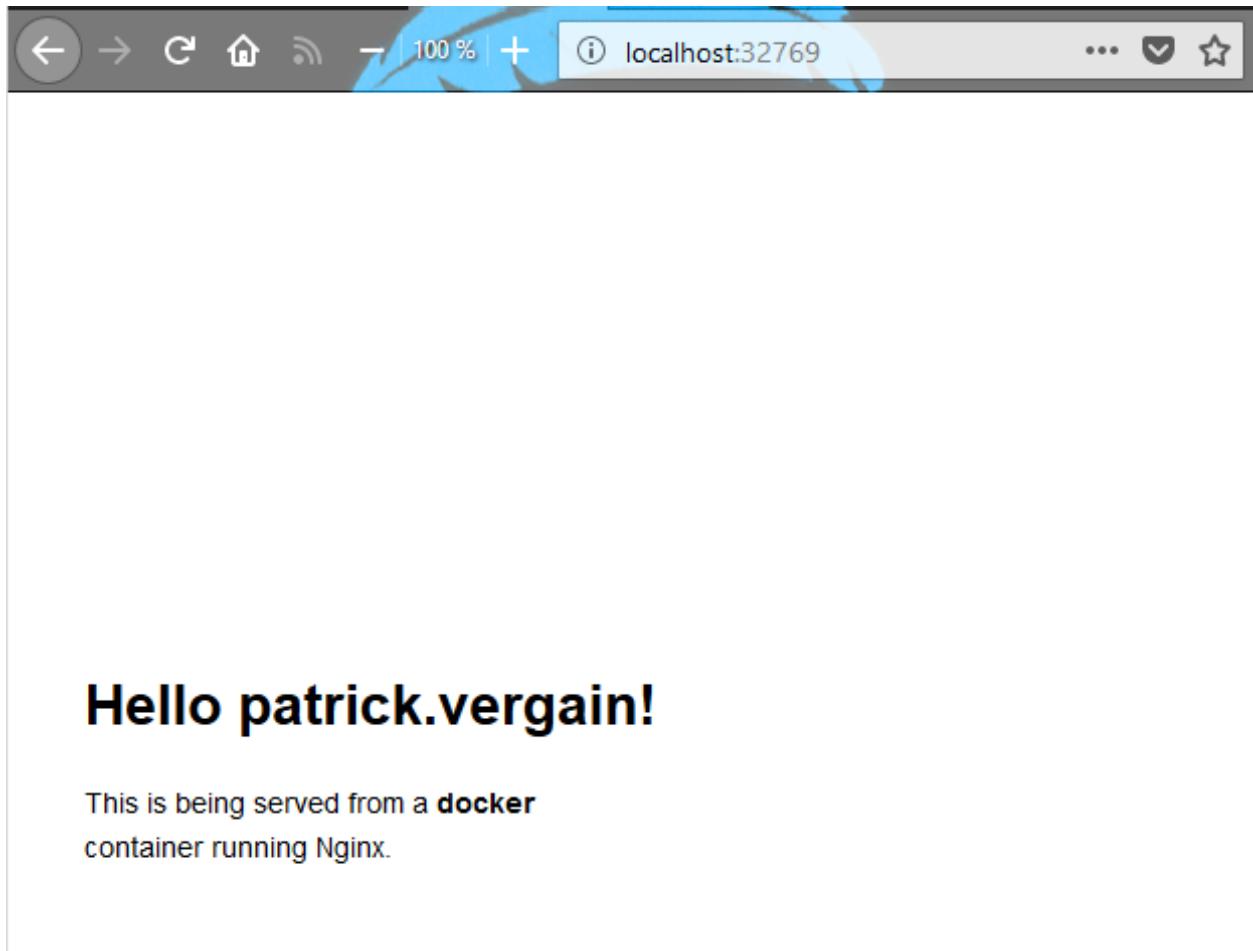
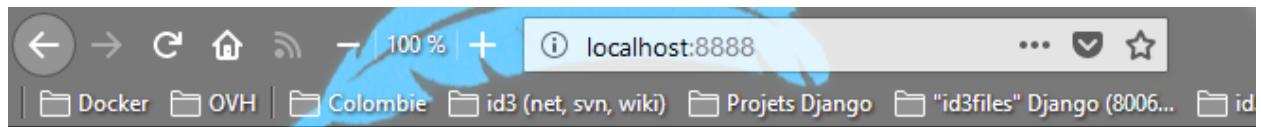


Fig. 1: <http://localhost:32769/>

```
839649f1be575ec442f9fe94d6957b0f218b63af3dfa8df989f413e86896d16
```



Hello patrick.vergain!

This is being served from a **docker** container running Nginx.

Fig. 2: <http://localhost:8888/>

To deploy this on a real server you would just need to install Docker, and run the above docker command(as in this case you can see the AUTHOR is Docker which we passed as an environment variable).

Now that you've seen how to run a webserver inside a Docker container, **how do you create your own Docker image ?**

This is the question we'll explore in the next section.

But first, let's stop and remove the containers since you won't be using them anymore.

25.1.2.7 docker stop static-site

```
docker stop static-site
```

```
static-site
```

25.1.2.8 docker rm static-site

```
docker rm static-site
```

```
static-site
```

25.1.2.9 Let's use a shortcut to remove the second site: docker rm -f static-site-2

```
docker rm -f static-site-2
```

```
static-site-2
```

25.1.2.10 Docker Images

See also:

- <http://linuxfr.org/news/sortie-d-ubuntu-16-04-lts-xenial-xerus>

In this section, let's dive deeper into what Docker images are.

You will build your own image, use that image to run an application locally, and finally, push some of your own images to Docker Cloud.

Docker images are the basis of containers. In the previous example, you pulled the dockersamples/static-site image from the registry and asked the Docker client to run a container based on that image.

To see the list of images that are available locally on your system, run the docker images command.

```
C:\Tmp>docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED
ubuntu	trusty	02a63d8b2bfa	18 hours ago
id3pvergain/get-started	part2	ed5b70620e49	32 hours ago
friendlyhello	latest	ed5b70620e49	32 hours ago
alpine	latest	3fd9065eaf02	6 days ago
wordpress	latest	28084cde273b	7 days ago
centos	latest	ff426288ea90	7 days ago
nginx	latest	3f8a4339aadd	2 weeks ago
ubuntu	latest	00fd29ccc6f1	4 weeks ago
python	2.7-slim	4fd30fc83117	5 weeks ago
hello-world	latest	f2a91732366c	8 weeks ago
docker4w/nsenter-dockerd	latest	cae870735e91	2 months ago
dockersamples/static-site	latest	f589ccde7957	22 months ago
↪ 191MB			(continues on next page)

(continued from previous page)

Above is a list of images that I've pulled from the registry and those I've created myself (we'll shortly see how). You will have a different list of images on your machine. The TAG refers to a particular snapshot of the image and the ID is the corresponding unique identifier for that image.

For simplicity, you can think of an image akin to a git repository - images can be committed with changes and have multiple versions. When you do not provide a specific version number, the client defaults to latest.

For example you could pull a specific version of ubuntu image as follows:

25.1.2.11 docker pull ubuntu:16.04

```
docker pull ubuntu:16.04
```

```
16.04: Pulling from library/ubuntu
8f7c85c2269a: Pull complete
9e72e494a6dd: Pull complete
3009ec50c887: Pull complete
9d5ffccbec91: Pull complete
e872a2642ce1: Pull complete
Digest: sha256:d3fdf5b1f8e8a155c17d5786280af1f5a04c10e95145a515279cf17abdf0191f
Status: Downloaded newer image for ubuntu:16.04
```

If you do not specify the version number of the image then, as mentioned, the Docker client will default to a version named latest.

So for example, the docker pull command given below will pull an image named ubuntu:latest:

```
docker pull ubuntu
```

To get a new Docker image you can either get it from a registry (such as the Docker Store) or create your own. There are hundreds of thousands of images available on Docker Store. You can also search for images directly from the command line using docker search.

An important distinction with regard to images is between base images and child images.

- Base images are images that have no parent images, usually images with an OS like ubuntu, alpine or debian.
- Child images are images that build on base images and add additional functionality.

Another key concept is the idea of official images and user images. (Both of which can be base images or child images.)

Official images are Docker sanctioned images. Docker, Inc. sponsors a dedicated team that is responsible for reviewing and publishing all Official Repositories content. This team works in collaboration with upstream software maintainers, security experts, and the broader Docker community.

These are not prefixed by an organization or user name. In the list of images above, the python, node, alpine and nginx images are official (base) images. To find out more about them, check out the Official Images Documentation.

User images are images created and shared by users like you. They build on base images and add additional functionality. Typically these are formatted as user/image-name. The user value in the image name is your Docker Store user or organization name.

25.1.2.12 Create your first image

Note: The code for this section is in this repository in the flask-app directory.

Now that you have a better understanding of images, it's time to create your own. Our goal here is to create an image that sandboxes a small Flask application.

The goal of this exercise is to create a Docker image which will run a Flask app.

We'll do this by first pulling together the components for a random cat picture generator built with Python Flask, then dockerizing it by writing a Dockerfile.

Finally, we'll build the image, and then run it.

- Create a Python Flask app that displays random cat pix
- Write a Dockerfile
- Build the image
- Run your image
- Dockerfile commands summary

25.1.2.13 Create a Python Flask app that displays random cat pix

For the purposes of this workshop, we've created a fun little Python Flask app that displays a random cat .gif every time it is loaded because, you know, who doesn't like cats ?

Start by creating a directory called flask-app where we'll create the following files:

- app.py
- requirements.txt
- templates/index.html
- Dockerfile

Make sure to cd flask-app before you start creating the files, because you don't want to start adding a whole bunch of other random files to your image.

25.1.2.13.1 app.py

Create the app.py with the following content.

```

1  """app.py
2
3
4
5 """
6
7 from flask import Flask, render_template
8 import random
9
10 app = Flask(__name__)
11
12 # list of cat images

```

(continues on next page)

(continued from previous page)

```

13 images = [
14     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr05/15/9/anigif_enhanced-
15     ↵buzz-26388-1381844103-11.gif",
16     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr01/15/9/anigif_enhanced-
17     ↵buzz-31540-1381844535-8.gif",
18     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr05/15/9/anigif_enhanced-
19     ↵buzz-26390-1381844163-18.gif",
20     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr06/15/10/anigif_enhanced-
21     ↵buzz-1376-1381846217-0.gif",
22     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr03/15/9/anigif_enhanced-
23     ↵buzz-3391-1381844336-26.gif",
24     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr06/15/10/anigif_enhanced-
25     ↵buzz-29111-1381845968-0.gif",
26     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr03/15/9/anigif_enhanced-
27     ↵buzz-3409-1381844582-13.gif",
28     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr02/15/9/anigif_enhanced-
29     ↵buzz-19667-1381844937-10.gif",
30     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr05/15/9/anigif_enhanced-
31     ↵buzz-26358-1381845043-13.gif",
32     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr06/15/9/anigif_enhanced-
33     ↵buzz-18774-1381844645-6.gif",
34     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr06/15/9/anigif_enhanced-
35     ↵buzz-25158-1381844793-0.gif",
36     "http://ak-hdl.buzzfed.com/static/2013-10/enhanced/webdr03/15/10/anigif_enhanced-
37     ↵buzz-11980-1381846269-1.gif"
38 ]
39
40 @app.route('/')
41 def index():
42     url = random.choice(images)
43     return render_template('index.html', url=url)
44
45 if __name__ == "__main__":
46     app.run(host="0.0.0.0")

```

25.1.2.13.2 requirements.txt

In order to install the Python modules required for our app, we need to create a file called requirements.txt and add the following line to that file

```
1 Flask==0.10.1
```

25.1.2.13.3 templates/index.html

Create a directory called templates and create an index.html file in that directory with the following content in it.

```

1 <html>
2   <head>
3     <style type="text/css">
4       body {
5         background: black;
6         color: white;
7       }

```

(continues on next page)

(continued from previous page)

```

8   div.container {
9     max-width: 500px;
10    margin: 100px auto;
11    border: 20px solid white;
12    padding: 10px;
13    text-align: center;
14  }
15  h4 {
16    text-transform: uppercase;
17  }
18  </style>
19  </head>
20  <body>
21    <div class="container">
22      <h4>Cat Gif of the day</h4>
23      
24      <p><small>Courtesy: <a href="http://www.buzzfeed.com/copyranter/the-best-cat-
25      ↪gif-post-in-the-history-of-cat-gifs">Buzzfeed</a></small></p>
26    </div>
27  </body>
</html>

```

25.1.2.13.4 Write a Dockerfile

See also:

- <https://docs.docker.com/engine/reference/builder/>

We want to create a Docker image with this web app. As mentioned above, all user images are based on a base image. Since our application is written in Python, we will build our own Python image based on *Alpine*. We'll do that using a Dockerfile.

A Dockerfile is a text file that contains a list of commands that the Docker daemon calls while creating an image. The Dockerfile contains all the information that Docker needs to know to run the app, a base Docker image to run from, location of your project code, any dependencies it has, and what commands to run at start-up. It is a simple way to automate the image creation process. The best part is that the commands you write in a Dockerfile are almost identical to their equivalent Linux commands. This means you don't really have to learn new syntax to create your own Dockerfiles.

FROM alpine:3.5

We'll start by specifying our base image, using the FROM keyword:

```
FROM alpine:3.5
```

RUN apk add –update py2-pip

The next step usually is to write the commands of copying the files and installing the dependencies. But first we will install the Python pip package to the alpine linux distribution. This will not just install the pip package but any other dependencies too, which includes the python interpreter. Add the following RUN command next:

```
RUN apk add --update py2-pip
```

Let's add the files that make up the Flask Application.

COPY requirements.txt /usr/src/app/

Install all Python requirements for our app to run. This will be accomplished by adding the lines:

```
COPY requirements.txt /usr/src/app/  
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
```

COPY app.py /usr/src/app/

Copy the files you have created earlier into our image by using COPY command.

```
COPY app.py /usr/src/app/  
COPY templates/index.html /usr/src/app/templates/
```

EXPOSE 5000

Specify the port number which needs to be exposed. Since our flask app is running on 5000 that's what we'll expose.

```
EXPOSE 5000
```

CMD ["python", "/usr/src/app/app.py"]

The last step is the command for running the application which is simply python ./app.py. Use the CMD command to do that:

```
CMD ["python", "/usr/src/app/app.py"]
```

The primary purpose of CMD is to tell the container which command it should run by default when it is started.

Verify your Dockerfile.

Our Dockerfile is now ready. This is how it looks:

```
1 # our base image  
2 FROM alpine:3.5  
3  
4 # Install python and pip  
5 RUN apk add --update py2-pip  
6  
7 # install Python modules needed by the Python app  
8 COPY requirements.txt /usr/src/app/  
9 RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt  
10  
11 # copy files required for the app to run  
12 COPY app.py /usr/src/app/  
13 COPY templates/index.html /usr/src/app/templates/  
14
```

(continues on next page)

(continued from previous page)

```

15 # tell the port number the container should expose
16 EXPOSE 5000
17
18 # run the application
19 CMD ["python", "/usr/src/app/app.py"]
```

25.1.2.13.5 Build the image (docker build -t id3pvergain/myfirstapp)

Now that you have your Dockerfile, you can build your image.

The docker build command does the heavy-lifting of creating a docker image from a Dockerfile.

When you run the docker build command given below, make sure to replace <YOUR_USERNAME> with your user-name.

This username should be the same one you created when registering on Docker Cloud. If you haven't done that yet, please go ahead and create an account.

The docker build command is quite simple - it takes an optional tag name with the -t flag, and the location of the directory containing the Dockerfile - the . indicates the current directory:

```
docker build -t id3pvergain/myfirstapp .
```

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
 ↵docker\samples\labs\webapps\app_flask>docker build -t id3pvergain/myfirstapp .
```

```

Sending build context to Docker daemon    7.68kB
Step 1/8 : FROM alpine:3.5
3.5: Pulling from library/alpine
550felbea624: Pull complete
Digest: sha256:9148d069e50eee519ec45e5683e56a1c217b61a52ed90eb77bdce674cc212f1e
Status: Downloaded newer image for alpine:3.5
--> 6c6084ed97e5
Step 2/8 : RUN apk add --update py2-pip
--> Running in 1fe5bd53d58d
fetch http://dl-cdn.alpinelinux.org/alpine/v3.5/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.5/community/x86_64/APKINDEX.tar.gz
(1/12) Installing libbz2 (1.0.6-r5)
(2/12) Installing expat (2.2.0-r1)
(3/12) Installing libffi (3.2.1-r2)
(4/12) Installing gdbm (1.12-r0)
(5/12) Installing ncurses-terminfo-base (6.0_p20170701-r0)
(6/12) Installing ncurses-terminfo (6.0_p20170701-r0)
(7/12) Installing ncurses-libs (6.0_p20170701-r0)
(8/12) Installing readline (6.3.008-r4)
(9/12) Installing sqlite-libs (3.15.2-r1)
(10/12) Installing python2 (2.7.13-r0)
(11/12) Installing py-setuptools (29.0.1-r0)
(12/12) Installing py2-pip (9.0.0-r1)
Executing busybox-1.25.1-r1.trigger
OK: 61 MiB in 23 packages
Removing intermediate container 1fe5bd53d58d
--> 23504d4e2c59
Step 3/8 : COPY requirements.txt /usr/src/app/
--> 1be30128b66f
```

(continues on next page)

(continued from previous page)

```

Step 4/8 : RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
--> Running in a5f6ada2483d
Collecting Flask==0.10.1 (from -r /usr/src/app/requirements.txt (line 1))
  Downloading Flask-0.10.1.tar.gz (544kB)
Collecting Werkzeug>=0.7 (from Flask==0.10.1->-r /usr/src/app/requirements.txt (line 1))
  Downloading Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
Collecting Jinja2>=2.4 (from Flask==0.10.1->-r /usr/src/app/requirements.txt (line 1))
  Downloading Jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting itsdangerous>=0.21 (from Flask==0.10.1->-r /usr/src/app/requirements.txt (line 1))
  Downloading itsdangerous-0.24.tar.gz (46kB)
Collecting MarkupSafe>=0.23 (from Jinja2>=2.4->Flask==0.10.1->-r /usr/src/app/requirements.txt (line 1))
  Downloading MarkupSafe-1.0.tar.gz
Installing collected packages: Werkzeug, MarkupSafe, Jinja2, itsdangerous, Flask
  Running setup.py install for MarkupSafe: started
    Running setup.py install for MarkupSafe: finished with status 'done'
  Running setup.py install for itsdangerous: started
    Running setup.py install for itsdangerous: finished with status 'done'
  Running setup.py install for Flask: started
    Running setup.py install for Flask: finished with status 'done'
Successfully installed Flask-0.10.1 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1
itsdangerous-0.24
You are using pip version 9.0.0, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Removing intermediate container a5f6ada2483d
--> 68467d64c546
Step 5/8 : COPY app.py /usr/src/app/
--> 62a6a857c6cd
Step 6/8 : COPY templates/index.html /usr/src/app/templates/
--> 639c61ea4a4b
Step 7/8 : EXPOSE 5000
--> Running in c15c0178577c
Removing intermediate container c15c0178577c
--> f6d0fdcd6c29
Step 8/8 : CMD ["python", "/usr/src/app/app.py"]
--> Running in 222f91658593
Removing intermediate container 222f91658593
--> 0ce3c7641c9a
Successfully built 0ce3c7641c9a
Successfully tagged id3pvergain/myfirstapp:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows
Docker host. All files and directories added to build context will have '-rwxr-xr-x
' permissions. It is recommended to double check and reset permissions for
sensitive files and directories.

```

If you don't have the alpine:3.5 image, the client will first pull the image and then create your image. Therefore, your output on running the command will look different from mine. If everything went well, your image should be ready!

25.1.2.13.6 docker images

Run docker images and see if your image (<YOUR_USERNAME>/myfirstapp) shows.

```

Y:projects_id3P5N001XLOGCA135_tutorial_dockertutorial_dockersampleslabswebappsapp_flask>docker
images

```

REPOSITORY	TAG	IMAGE ID	CREATED	
id3pvergain/myfirstapp	latest	0ce3c7641c9a	2 minutes ago	[link]
ubuntu	16.04	2a4cca5ac898	38 hours ago	[link]
ubuntu	trusty	02a63d8b2bfa	38 hours ago	[link]
friendlyhello	latest	ed5b70620e49	2 days ago	[link]
id3pvergain/get-started	part2	ed5b70620e49	2 days ago	[link]
alpine	3.5	6c6084ed97e5	7 days ago	[link]
alpine	latest	3fd9065eaf02	7 days ago	[link]
wordpress	latest	28084cde273b	8 days ago	[link]
centos	latest	ff426288ea90	8 days ago	[link]
nginx	latest	3f8a4339aadd	3 weeks ago	[link]
ubuntu	latest	00fd29ccc6f1	4 weeks ago	[link]
python	2.7-slim	4fd30fc83117	5 weeks ago	[link]
hello-world	latest	f2a91732366c	8 weeks ago	[link]
docker4w/nsenter-dockerd	latest	cae870735e91	2 months ago	[link]
dockersamples/static-site	latest	f589ccde7957	22 months ago	[link]

25.1.2.13.7 Run your image (docker run -p 8888:5000 --name myfirstapp id3pvergain/myfirstapp)

The next step in this section is to run the image and see if it actually works.

```
docker run -p 8888:5000 --name myfirstapp id3pvergain/myfirstapp

* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Head over to <http://localhost:8888> and your app should be live.

Note: If you are using Docker Machine, you may need to open up another terminal and determine the container ip address using docker-machine ip default.

Hit the Refresh button in the web browser to see a few more cat images.

25.1.2.13.8 Push your image (docker push id3pvergain/myfirstapp)

Now that you've created and tested your image, you can push it to Docker Cloud.

First you have to login to your Docker Cloud account, to do that:

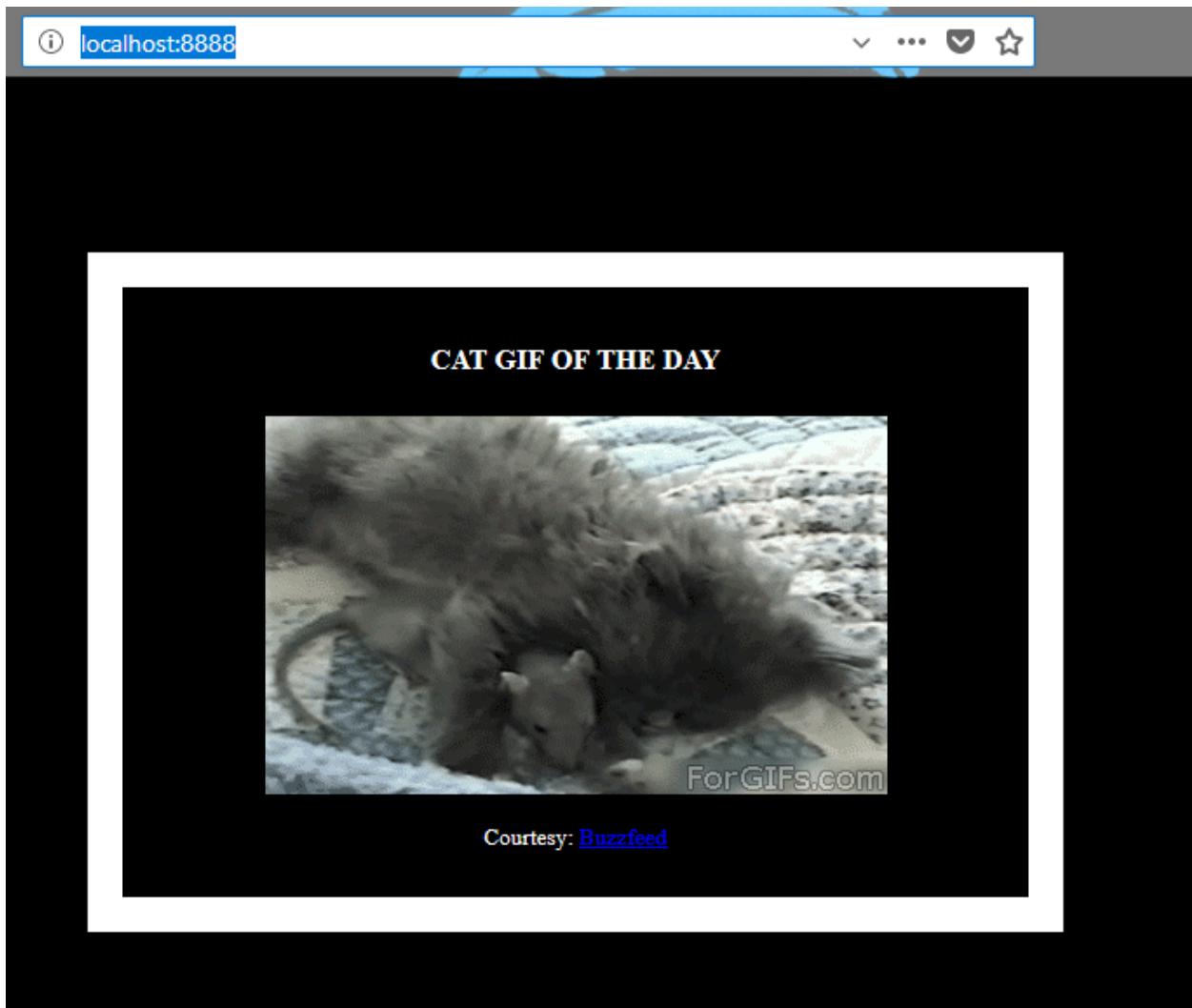


Fig. 3: <http://localhost:8888/>

docker login

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
↳ docker\samples\labs\webapps\app_flask>docker login
```

```
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username (id3pvergain):
Password:
Login Succeeded
```

docker push id3pvergain/myfirstapp

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
↳ docker\samples\labs\webapps\app_flask>docker push id3pvergain/myfirstapp
```

```
The push refers to repository [docker.io/id3pvergain/myfirstapp]
b7591dd05809: Pushed
cd36128c70d4: Pushed
cea459424f6e: Pushed
6ac80674ef6a: Pushed
de7b45529bcb: Pushed
d39d92664027: Mounted from library/alpine
latest: digest: sha256:8f945ed63e2dc3ef3fa178fe4dded5a68eae07c5c9e854ec278c7cfa2c6bc6bb size: 1572
```

25.1.2.13.9 docker rm -f myfirstapp

Now that you are done with this container, stop and remove it since you won't be using it again.

Open another terminal window and execute the following commands:

```
docker stop myfirstapp
docker rm myfirstapp
```

or:

```
docker rm -f myfirstapp
```

```
myfirstapp
```

25.1.2.13.10 docker ps

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
↳ docker\samples\labs\webapps\app_flask>docker ps
```

CONTAINER ID ↳ STATUS	IMAGE PORTS	COMMAND NAMES	CREATED
--------------------------	----------------	------------------	---------

25.1.2.14 Dockerfile commands summary

Here's a quick summary of the few basic commands we used in our Dockerfile.

25.1.2.14.1 FROM

FROM starts the Dockerfile. It is a requirement that the Dockerfile must start with the FROM command. Images are created in layers, which means you can use another image as the base image for your own. The FROM command defines your base layer. As arguments, it takes the name of the image. Optionally, you can add the Docker Cloud username of the maintainer and image version, in the format `username/imagename:version`.

25.1.2.14.2 RUN

RUN is used to build up the Image you're creating. For each RUN command, Docker will run the command then create a new layer of the image. This way you can roll back your image to previous states easily. The syntax for a RUN instruction is to place the full text of the shell command after the RUN (e.g., RUN mkdir /user/local/foo). This will automatically run in a /bin/sh shell. You can define a different shell like this: RUN /bin/bash -c 'mkdir /user/local/foo'

25.1.2.14.3 COPY

COPY copies local files into the container.

25.1.2.14.4 CMD

CMD defines the commands that will run on the Image at start-up.

Unlike a RUN, *this does not create a new layer for the Image*, but simply runs the command.

There can only be one CMD per a Dockerfile/Image.

If you need to run multiple commands, the best way to do that is to have the CMD run a script. CMD requires that you tell it where to run the command, unlike RUN.

So example CMD commands would be:

```
CMD ["python", "./app.py"]
CMD ["/bin/bash", "echo", "Hello World"]
```

25.1.2.14.5 EXPOSE

EXPOSE creates a hint for users of an image which ports provide services. It is included in the information which can be retrieved via docker inspect <container-id>.

Note: The EXPOSE command does not actually make any ports accessible to the host! Instead, this requires publishing ports by means of the -p flag when using \$ docker run.

25.1.2.14.6 PUSH

PUSH pushes your image to Docker Cloud, or alternately to a private registry

Note: If you want to learn more about Dockerfiles, check out Best practices for writing Dockerfiles.

25.1.2.15 Next Steps : Deploying an app to a Swarm

See also:

- [3.0\) Deploying an app to a Swarm](#)

For the next step in the tutorial head over to [3.0 Deploying an app to a Swarm](#)

25.1.3 3.0) Deploying an app to a Swarm

See also:

- <https://github.com/docker/labs/blob/master/beginner/chapters/votingapp.md>
- <https://github.com/dockersamples/example-voting-app>
- [2\) Webapps with Docker \(Python + Flask\)](#)

Contents

- [3.0\) Deploying an app to a Swarm](#)
 - [Introduction](#)
 - [Voting app](#)
 - [Deploying the app](#)
 - * [docker swarm init](#)
 - * [Docker compose file : docker-stack.yml](#)
 - * [docker stack deploy –compose-file docker-stack.yml vote](#)
 - * [docker stack services vote](#)
 - * [Analyse du fichier Docker compose file : docker-stack.yml](#)
 - [compose-file: “3”](#)
 - [compose-file: services](#)
 - [compose-file: image](#)
 - [compose-file: ports and networks depends_on](#)
 - [compose-file: deploy](#)
 - [Test run : http://localhost:5000/](#)
 - [Customize the app](#)
 - * [Change the images used](#)

- * *Redeploy: docker stack deploy –compose-file docker-stack.yml vote*
 - * *Another test run*
 - * *Remove the stack*
- *Next steps*

25.1.3.1 Introduction

This portion of the tutorial will guide you through the creation and customization of a voting app. It's important that you follow the steps in order, and make sure to customize the portions that are customizable.

Warning: To complete this section, you will need to have Docker installed on your machine as mentioned in the Setup section. You'll also need to have git installed. There are many options for installing it. For instance, you can get it from GitHub.

25.1.3.2 Voting app

For this application we will use the Docker Example Voting App.

This app consists of five components:

- Python webapp which lets you vote between two options
- Redis queue which collects new votes
- DotNET worker which consumes votes and stores them in
- Postgres database backed by a Docker volume
- Node.js webapp which shows the results of the voting in real time

Clone the repository onto your machine and cd into the directory:

```
git clone https://github.com/docker/example-voting-app.git
```

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
↳ docker\samples\labs\votingapp>git clone https://github.com/docker/example-voting-
↳ app.git
```

```
Cloning into 'example-voting-app'...
remote: Counting objects: 463, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 463 (delta 4), reused 12 (delta 4), pack-reused 447
Receiving objects: 100% (463/463), 226.49 KiB | 318.00 KiB/s, done.
Resolving deltas: 100% (167/167), done.
```

```
cd example-voting-app
```

25.1.3.3 Deploying the app

See also:

- <https://docs.docker.com/engine/swarm/>

For this first stage, we will use existing images that are in Docker Store.

This app relies on [Docker Swarm mode](#). Swarm mode is the cluster management and orchestration features embedded in the Docker engine. You can easily deploy to a swarm using a file that declares your desired state for the app.

Swarm allows you to run your containers on more than one machine.

In this tutorial, *you can run on just one machine*, or you can use something like Docker for AWS or Docker for Azure to quickly create a multiple node machine. Alternately, you can use Docker Machine to create a number of local nodes on your development machine. See the Swarm Mode lab for more information.

25.1.3.3.1 docker swarm init

First, create a Swarm.

```
docker swarm init
```

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
↳docker\samples\labs\votingapp\example-voting-app>docker swarm init
```

```
Swarm initialized: current node (pfx5nyrmtv0m5twcz4dv4oypg) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-
↳1a5pls76a0tyfn9tybruku4naqaalvldvw0iy76hw9t6uw931w-098lzb69ozqce3v6eiptieeta 192.
↳168.65.3:2377
```

```
To add a manager to this swarm, run 'docker swarm join-token manager' and follow the
↳instructions.
```

Next, you will need a Docker Compose file. You don't need Docker Compose installed, though if you are using Docker for Mac or Docker for Windows you have it installed. However, docker stack deploy accepts a file in the Docker Compose format. The file you need is in Docker Example Voting App at the root level. It's called docker-stack.yml.

25.1.3.3.2 Docker compose file : docker-stack.yml

See also:

- <https://github.com/dockersamples/example-voting-app/blob/master/docker-stack.yml>

```
version: "3"
services:

  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - frontend
    deploy:
      replicas: 1
      update_config:
        parallelism: 2
        delay: 10s
```

(continues on next page)

(continued from previous page)

```

        restart_policy:
            condition: on-failure
db:
    image: postgres:9.4
    volumes:
        - db-data:/var/lib/postgresql/data
    networks:
        - backend
    deploy:
        placement:
            constraints: [node.role == manager]
vote:
    image: dockersamples/examplevotingapp_vote:before
    ports:
        - 5000:80
    networks:
        - frontend
    depends_on:
        - redis
    deploy:
        replicas: 2
        update_config:
            parallelism: 2
        restart_policy:
            condition: on-failure
result:
    image: dockersamples/examplevotingapp_result:before
    ports:
        - 5001:80
    networks:
        - backend
    depends_on:
        - db
    deploy:
        replicas: 1
        update_config:
            parallelism: 2
            delay: 10s
        restart_policy:
            condition: on-failure

worker:
    image: dockersamples/examplevotingapp_worker
    networks:
        - frontend
        - backend
    deploy:
        mode: replicated
        replicas: 1
        labels: [APP=VOTING]
        restart_policy:
            condition: on-failure
            delay: 10s
            max_attempts: 3
            window: 120s
        placement:
            constraints: [node.role == manager]

```

(continues on next page)

(continued from previous page)

```

visualizer:
  image: dockersamples/visualizer:stable
  ports:
    - "8080:8080"
  stop_grace_period: 1m30s
  volumes:
    - "/var/run/docker.sock:/var/run/docker.sock"
  deploy:
    placement:
      constraints: [node.role == manager]

networks:
  frontend:
  backend:

volumes:
  db-data:

```

25.1.3.3.3 docker stack deploy --compose-file docker-stack.yml vote

First deploy it, and then we will look more deeply into the details:

```

Y:\projects_id3\P5N001\xLOGCA135_tutorial_docker\tutorial_
  ↵docker\samples\labs\votingapp\example-voting-app>docker stack deploy --compose-file_
  ↵docker-stack.yml vote

```

```

Creating network vote_backend
Creating network vote_default
Creating network vote_frontend
Creating service vote_visualizer
Creating service vote_redis
Creating service vote_db
Creating service vote_vote
Creating service vote_result
Creating service vote_worker

```

25.1.3.3.4 docker stack services vote

to verify your stack has deployed, use *docker stack services vote*

```

Y:\projects_id3\P5N001\xLOGCA135_tutorial_docker\tutorial_
  ↵docker\samples\labs\votingapp\example-voting-app>docker stack services vote

```

ID	NAME	MODE	REPLICAS	IMAGE
PORTS				
d7ovptjpvv3y	vote_vote	replicated	0/2	
				dockersamples/examplevotingapp_vote:before *:5000->80/tcp
lve7cp7gxvg	vote_result	replicated	0/1	
				dockersamples/examplevotingapp_result:before *:5001->80/tcp
r2mhivfbyaua	vote_redis	replicated	1/1	
				*:30000->6379/tcp

(continues on next page)

(continued from previous page)

szzocr20dyfc	vote_visualizer	replicated	1/1	
↳dockersamples/visualizer:stable		* :8080→8080/tcp		✗
vgv0iucy6fx9	vote_db	replicated	0/1	✗
↳postgres:9.4				✗
vlieeu7ru24a	vote_worker	replicated	0/1	✗
↳dockersamples/examplevotingapp_worker:latest				✗

25.1.3.3.5 Analyse du fichier Docker compose file : docker-stack.yml

See also:

- <https://github.com/docker/labs/tree/master/networking>

If you take a look at docker-stack.yml, you will see that the file defines

- vote container based on a Python image
- result container based on a Node.js image
- redis container based on a redis image, to temporarily store the data.
- DotNET based worker app based on a .NET image
- Postgres container based on a postgres image

The Compose file also defines two networks, front-tier and back-tier.

Each container is placed on one or two networks.

Once on those networks, they can access other services on that network in code just by using the name of the service.

Services can be on any number of networks.

Services are isolated on their network.

Services are only able to discover each other by name if they are on the same network.

To learn more about networking check out the [Networking Lab](#).

compose-file: “3”

See also:

- <https://docs.docker.com/compose/compose-file/>

Take a look at the file again. You’ll see it starts with:

```
version: "3"
```

It’s important that you use **version 3** of compose files, as docker stack deploy won’t support use of earlier versions.

compose-file: services

You will see there’s also a **services** key, under which there is a separate key for each of the services. Such as:

```

vote:
  image: dockersamples/examplevotingapp_vote:before
  ports:
    - 5000:80
  networks:
    - frontend
  depends_on:
    - redis
  deploy:
    replicas: 2
    update_config:
      parallelism: 2
    restart_policy:
      condition: on-failure

```

compose-file: *image*

See also:

<https://docs.docker.com/compose/compose-file/#image>

The image key there specifies which image you can use, in this case the image dockersamples/examplevotingapp_vote:before.

If you're familiar with Compose, you may know that there's a build key, which builds based on a Dockerfile.

However, docker stack deploy does not support build, so you need to use pre-built images.

compose-file: *ports and networks depends_on*

See also:

- <https://docs.docker.com/compose/compose-file/#ports>

Much like docker run you will see you can define ports and networks.

There's also a depends_on key which allows you to specify that a service is only deployed after another service, in this case vote only deploys after redis.

compose-file: *deploy*

See also:

<https://docs.docker.com/compose/compose-file/#deploy>

The deploy key is new in version 3.

It allows you to specify various properties of the deployment to the Swarm.

In this case, you are specifying that you want two replicas, that is two containers are deployed on the Swarm. You can specify other properties, like when to restart, what healthcheck to use, placement constraints, resources.

Test run : <http://localhost:5000/>

Now that the app is running, you can go to <http://localhost:5000> to see:

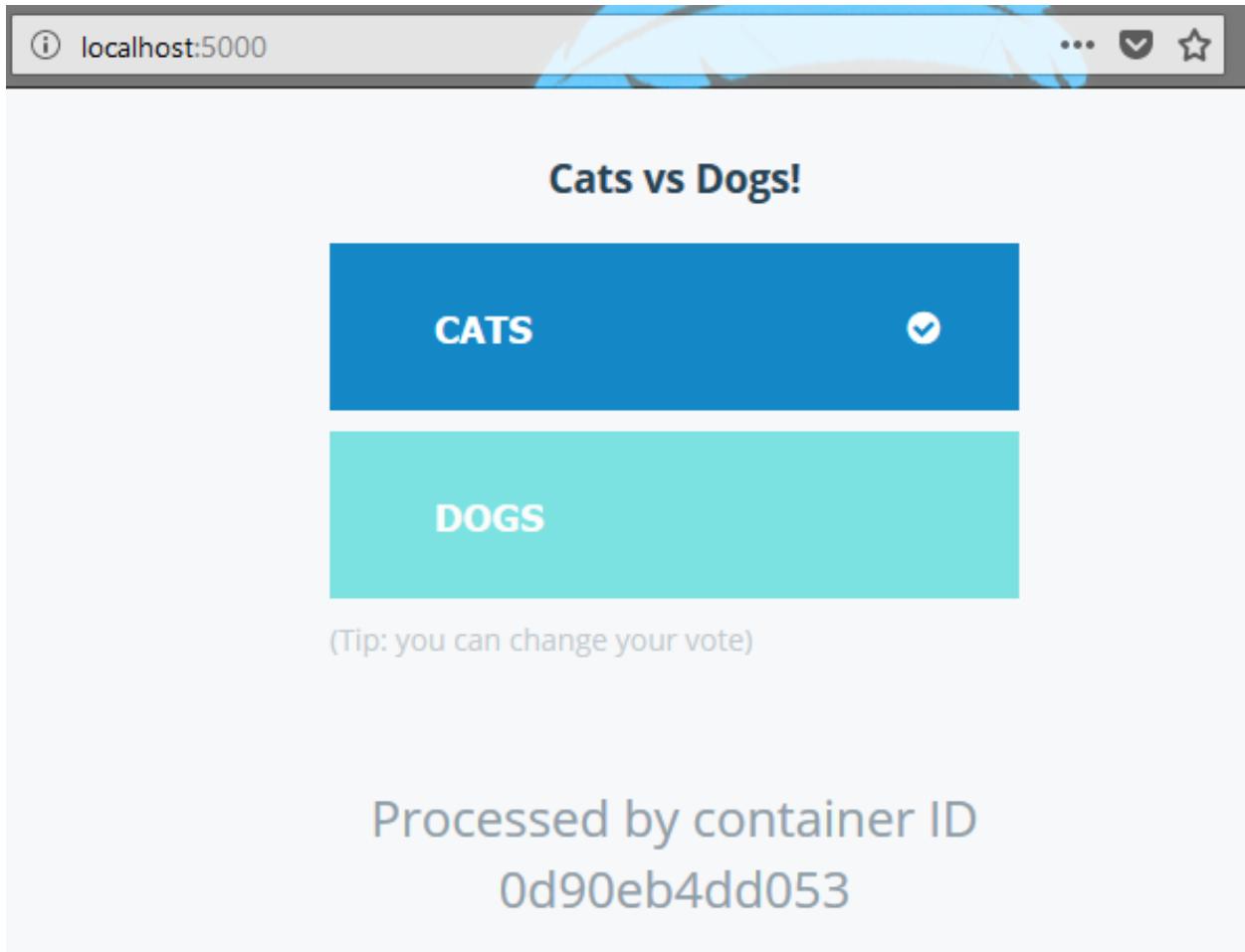


Fig. 4: <http://localhost:5000/>

Click on one to vote. You can check the results at <http://localhost:5001>

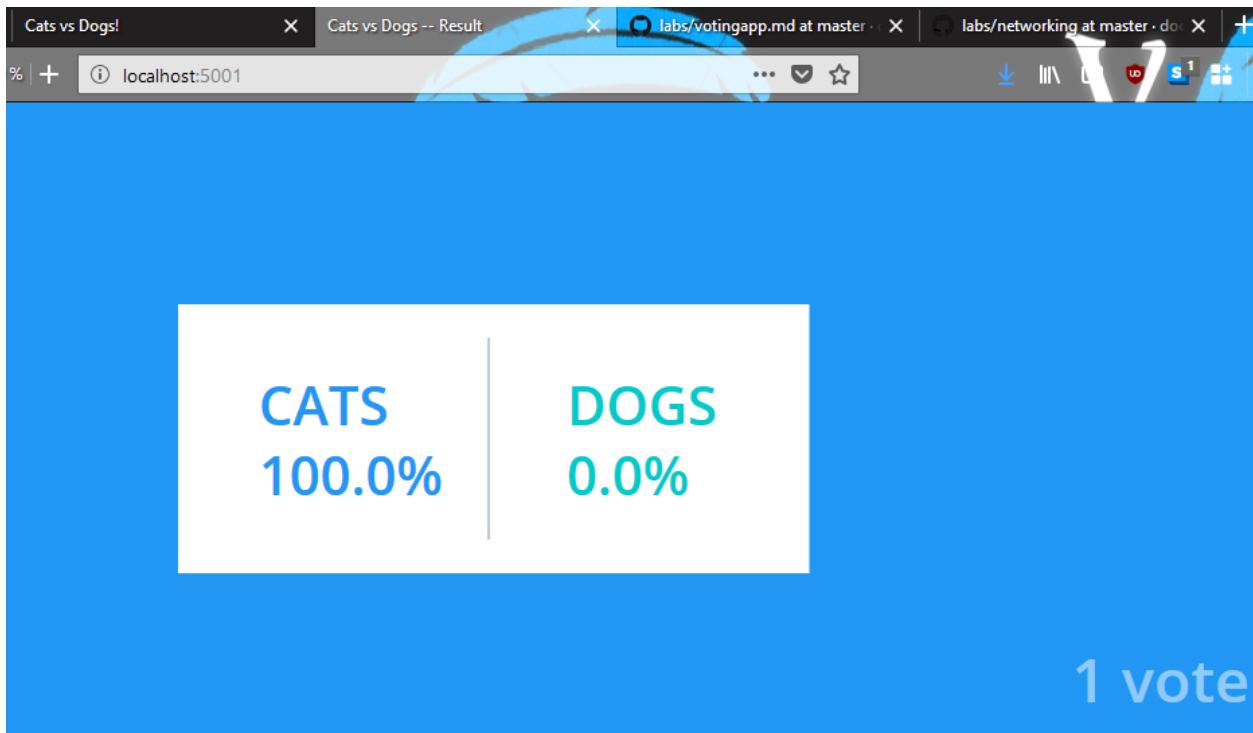


Fig. 5: <http://localhost:5001/>

Note: If you are running this tutorial in a cloud environment like AWS, Azure, Digital Ocean, or GCE you will not have direct access to localhost or 127.0.0.1 via a browser. A work around for this is to leverage ssh port forwarding.

Below is an example for Mac OS. Similarly this can be done for Windows and Putty users:

```
ssh -L 5000:localhost:5000 <ssh-user>@<CLOUD_INSTANCE_IP_ADDRESS>
```

25.1.3.4 Customize the app

In this step, you will customize the app and redeploy it.

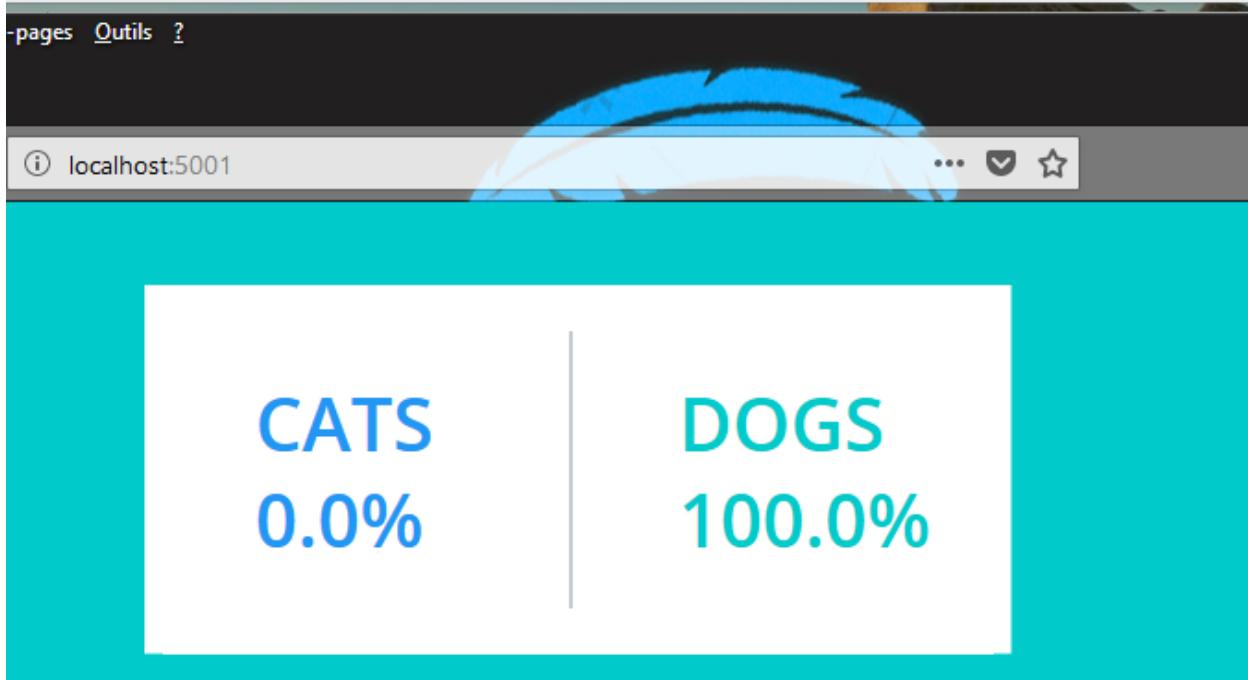
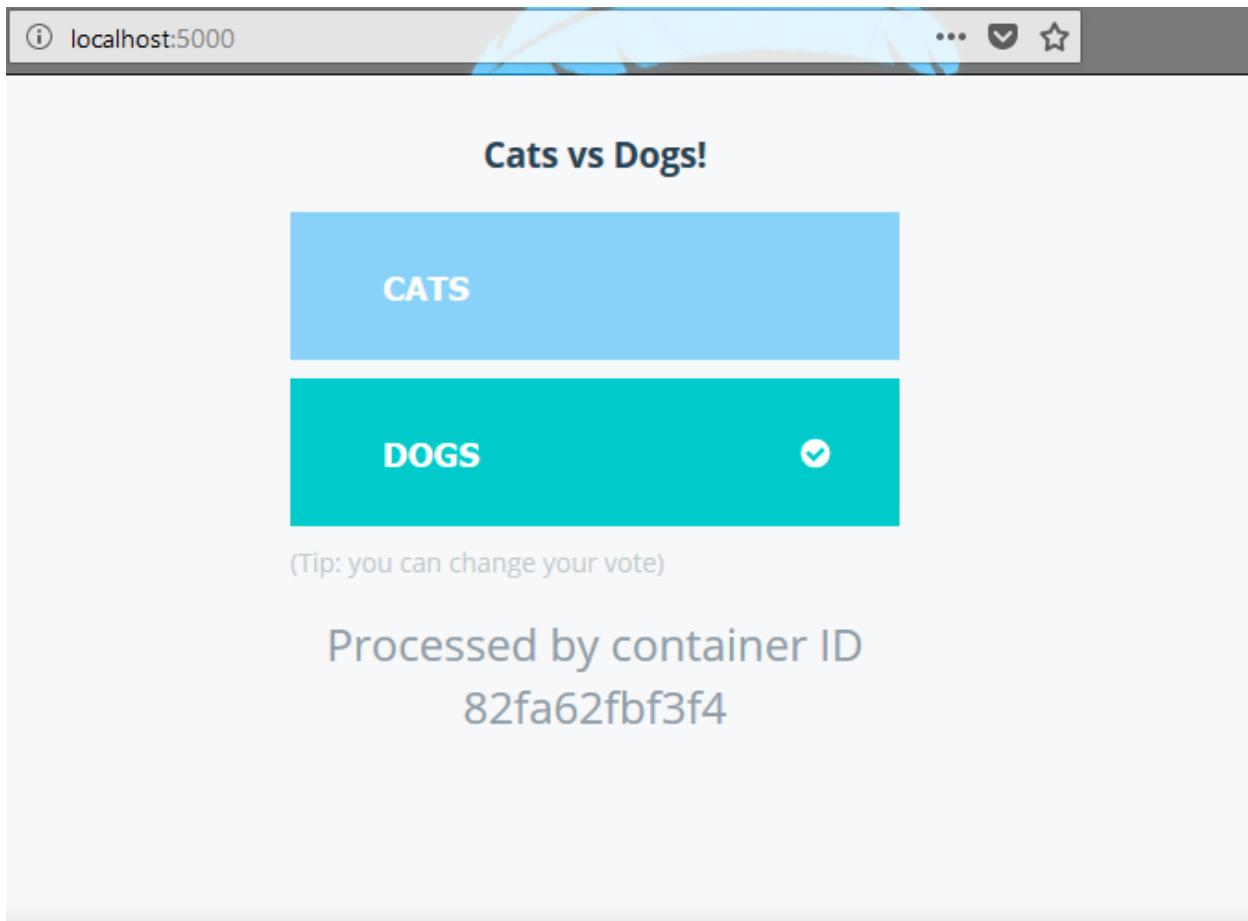
We've supplied the same images but with the votes changed from Cats and Dogs to Java and .NET using the after tag.

25.1.3.4.1 Change the images used

Going back to docker-stack.yml, change the vote and result images to use the after tag, so they look like this:

```
vote:
  image: dockersamples/examplevotingapp_vote:after
  ports:
    - 5000:80
networks:
  - frontend
```

(continues on next page)



(continued from previous page)

```

depends_on:
  - redis
deploy:
  replicas: 2
  update_config:
    parallelism: 2
  restart_policy:
    condition: on-failure
result:
  image: dockersamples/examplevotingapp_result:after
  ports:
  - 5001:80
networks:
  - backend
depends_on:
  - db
deploy:
  replicas: 2
  update_config:
    parallelism: 2
    delay: 10s
  restart_policy:
    condition: on-failure

```

25.1.3.4.2 Redeploy: docker stack deploy --compose-file docker-stack.yml vote

Redeployment is the same as deploying:

```
docker stack deploy --compose-file docker-stack.yml vote
```

```
Y:\projects_id3\P5N001\XLOGCA135_tutorial_docker\tutorial_
→docker\samples\labs\votingapp\example-voting-app>docker stack deploy --compose-file_
→docker-stack.yml vote
```

```

Updating service vote_db (id: vgv0iucy6fx9ih4so6ufdzqh4)
Updating service vote_vote (id: d7ovptjpvv3ylpxb30hitxd1g)
Updating service vote_result (id: lve7cp7gxvwge1qhesjwuyon1)
Updating service vote_worker (id: vlieeu7ru24a8kc4vouwa0i5r)
Updating service vote_visualizer (id: szzocr20dyfc6ux0vdnamo5e1)
Updating service vote_redis (id: r2mhivfbayaunnd5szq5kh5fm7)

```

25.1.3.4.3 Another test run

Now take it for a spin again. Go to the URLs you used in section 3.1 and see the new votes.

25.1.3.4.4 Remove the stack

Remove the stack from the swarm:

```
docker stack rm vote
```

```

Y:\projects_id3\PSN001\XLOGCA135_tutorial_docker\tutorial_docker\samples\labs\votingapp\example-voting-app>docker swarm init
Swarm initialized: current node (pxf5nyrmvbm5tvcz4dv40ypg) is now a manager.

To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-1a5pls76a0tyfn9tybruku4naqa1vlvdw0iy70hw9t6uw931w-098l2v69ozqce3v6eiptieeta 192.168.65.3:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

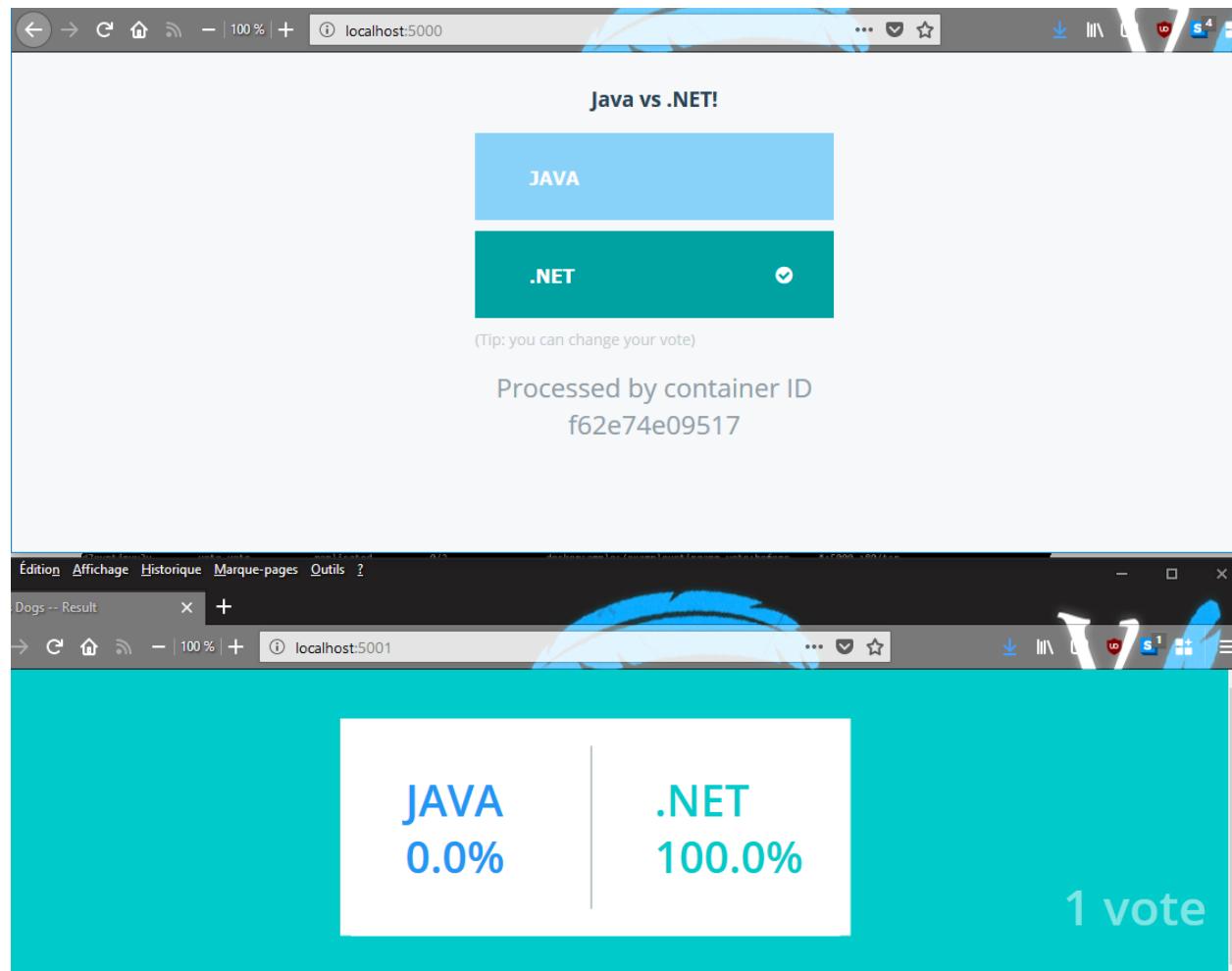
Y:\projects_id3\PSN001\XLOGCA135_tutorial_docker\tutorial_docker\samples\labs\votingapp\example-voting-app>docker stack deploy --compose-file docker-stack.yml vote
Creating network vote_backend
Creating network vote_default
Creating network vote_frontend
Creating service vote_visualizer
Creating service vote_redis
Creating service vote_db
Creating service vote_vote
Creating service vote_result
Creating service vote_worker

Y:\projects_id3\PSN001\XLOGCA135_tutorial_docker\tutorial_docker\samples\labs\votingapp\example-voting-app>docker stack services vote
ID          NAME      MODE        REPLICAS   IMAGE
d7ovptjpvv3y  vote_vote  replicated  0/2       dockersamples/examplevotingapp_vote:before
                                                PORTS
                                                *:5000->80/tcp
lve7cp7gxwg  vote_result  replicated  0/1      dockersamples/examplevotingapp_result:before
                                                *:5001->80/tcp
r2mhivfbayun  vote_redis  replicated  1/1      redis:alpine
                                                *:30000->6379/tcp
szzocr20dyfc  vote_visualizer  replicated  1/1    dockersamples/visualizer:stable
                                                *:8080->8080/tcp
vgv0iucy6fx9  vote_db    replicated  0/1      postgres:9.4
vlieeu7ru24a  vote_worker  replicated  0/1      dockersamples/examplevotingapp_worker:latest

Y:\projects_id3\PSN001\XLOGCA135_tutorial_docker\tutorial_docker\samples\labs\votingapp\example-voting-app>docker stack deploy --compose-file docker-stack.yml vote
Updating service vote_db (id: vgv0iucy6fx9ih4sodufdzqn4)
Updating service vote_vote (id: d7ovptjpvv3ylpxb30h1txdig)
Updating service vote_result (id: lve7cp7gxwgelqhesjwyon1)
Updating service vote_worker (id: vlieeu7ru24akc4vouwa015r)
Updating service vote_visualizer (id: szzocr20dyfc6uxv9dmamo5e1)
Updating service vote_redis (id: r2mhivfbayundsszqskh5fm7)

Y:\projects_id3\PSN001\XLOGCA135_tutorial_docker\tutorial_docker\samples\labs\votingapp\example-voting-app>_

```



```
Removing service vote_db
Removing service vote_redis
Removing service vote_result
Removing service vote_visualizer
Removing service vote_vote
Removing service vote_worker
Removing network vote_frontend
Removing network vote_default
Removing network vote_backend
```

25.1.3.5 Next steps

See also:

- <https://docs.docker.com/>
- <https://forums.docker.com/>
- <https://stackoverflow.com/tags/docker/>

Now that you've built some images and pushed them to Docker Cloud, and learned the basics of Swarm mode, you can explore more of Docker by checking out the [documentation](#).

And if you need any help, check out the Docker [Forums](#) or [StackOverflow](#).

25.2 Exemples sur Windows 10

See also:

- <https://docs.microsoft.com/fr-fr/virtualization/windowscontainers/quick-start/quick-start-windows-10>

Index

Symbols

- _container
 - restarting, 194
 - 1.21.2 (2018-05-02)
 - docker-compose, 34
 - 1.22 (2018-07-18)
 - docker-compose, 34
 - 17.06.0-ce (2017-06-23, 02c1d87)
 - docker-ce, 24
 - 17.12.1-ce (2018-02-27)
 - docker-ce, 24
 - 18.03.1-ce (2018-04-26, 9ee9f40)
 - docker-ce, 24
 - 18.06.0-ce (2018-07-18)
 - docker-ce, 24
 - 18.06.1-ce (2018-08-21)
 - docker-ce, 24
 - 18.09-ce (2018-11-08)
 - docker-ce, 24
 - 2017-08
 - News, 125
 - 2018-01
 - News, 118
 - 2018-01-29
 - News, 124, 281
 - 2018-01-30
 - News, 123
 - 2018-01-31
 - News, 119
 - 2018-02
 - News, 110
 - 2018-02-12
 - Action, 117
 - 2018-02-13
 - Action, 115
 - 2018-03
 - News, 110
 - 2018-03-29
 - Jérôme Petazzoni, 110
- 2018-04
 - News, 109
 - 2018-05
 - Lacey Williams, 107
 - Mickael Baron, 107
 - News, 107
 - 2018-06
 - docker-compose.yml, 106
 - Dockerfiles, 106
 - Lacey Williams, 106
 - News, 106
 - 2018-08
 - News, 105
 - 2018-09
 - News, 105
 - 2018-11
 - News, 105
 - 2018-11-08
 - News, 105
 - 3.6
 - compose-file, 48
 - 9-pillars-of-containers-best-practices
 - Best practices, 52
- ### Numbers
- 2017
 - News, 125
 - 2018
 - News, 105
- ### A
- Action
 - 2018-02-12, 117
 - 2018-02-13, 115
 - Adam
 - King, 281
 - Adam King
 - Dockerfile, 281
 - Agile Software Development, 85
 - Agilité

Définitions, 1
Alpine
 Image, 131
Ambassadors
 Container, 242
AMQP
 rabbitmq, 169
Anaconda3
 Images, 173
Apache HTTP Bitnami server
 Image, 150
Apache HTTP server
 Images, 148
Apache Tomcat
 Images, 157
Application
 Swarm, 417
Apps
 Web, 399
Aquasec
 Docker, 93
 Security, 93
Articles
 Docker Swarm, 61
Articles (2018)
 Docker Swarm, 61
attach
 docker, 63
attaching
 container, 194
Avril 2018
 Chapter1, 188
 Chapter5, 242
 Chapter6, 263
 Chapter7, 263
 Chapter8, 263
 History, 184
 Intro, 177
 Jérôme Petazzoni, 177
 Overview, 177
 Tutoriel, 177

B

background
 container, 188, 189
Best practices
 9-pillars-of-containers-best-practices, 52
 Docker, 52
 Dockerfiles, 50
 Nick Janetakis, 95
Beuret
 Stéphane, 97
Bonnes pratiques
 Docker, 50

Bret
 Fischer, 95
Bret Fischer
 People, 95
Bright
 Mickael, 97
build
 docker, 63
 docker-compose, 26
build –no-cache
 docker, 63
Building
 Images, 204
builds
 Multi-stage, 223

C

celery
 rabbitmq, 169
CentOS, 136
Centos
 Image, 136
Centos7
 Tutoriel, 321
Chapter1
 Avril 2018, 188
Chapter5
 Avril 2018, 242
Chapter6
 Avril 2018, 263
Chapter7
 Avril 2018, 263
Chapter8
 Avril 2018, 263
class
 image, 196
CMD
 Dockerfile, 215
 Overriding, 215
Collation
 <https://docs.postgresql.fr/10/charset.html>, 368
commands
 docker, 63
 docker-compose, 26
commit
 docker, 63
Compose
 Django, 34
 Docker, 24
 gitlab ARM, 47
compose
 django, 44
 docker, 252
Compose file

Examples, 44
compose-file, 48
 3.6, 48
Container
 Ambassadors, 242
 Inside, 99, 236
 Network drivers, 241
 Network model, 241
 Networking basics, 240
 Registry, 82
 Service discovery, 241
container
 attaching, 194
 background, 188, 189
 detach-keys, 194
 detaching, 194
 instance, 196
 logs, 188, 189
 non-interactive, 188
container ;
 shell, 87
Containers, 85
 Labels, 234
 Naming, 231
containers
 first, 188
 IDs, 188, 189
 restarting, 194
 running, 188, 189
Cook
 Jacob, 286
cookiecutter-django
 Docker, 106
COPY
 Dockerfile, 220
copying
 files, 220
cp
 docker, 63
Création
 PostgreSQL, 110

D

Définitions
 Agilité, 1
 Devops, 1
 Docker, 1
Database
 db_id3_intranet, 115
db_id3_intranet
 Database, 115
Debian, 132
 Image, 132
deprecated

MAINTAINER, 74
detach
 sequence, 194
detach-keys
 container, 194
detaching
 container, 194
Devops
 Définitions, 1
diff
 docker, 63
Django
 Compose, 34
 Docker, 313
 Jeff Triplett, 106
 Joe Jasinski, 106
djangox
 compose, 44
Django (erroneousboat)
 Docker, 314
Django (for beginners)
 Docker, 291
djangox
 Docker, 291
Doc
 Docker, 91
Doc (Aquasec)
 Docker, 93
Docker, 85
 Aquasec, 93
 Best practices, 52
 Bonnes pratiques, 50
 Compose, 24
 cookiecutter-django, 106
 Définitions, 1
 Django, 313
 Django (erroneousboat), 314
 Django (for beginners), 291
 djangox, 291
 Doc, 91
 Doc (Aquasec), 93
 docker-compose.yml, 368
 Dockerfile, 74
 Hébergeurs, 90
 hub, 226
 Images, 126
 Installation, 15
 Introduction, 1
 Jérôme, 263
 Jeff Triplett, 106
 Joe Jasinski, 106
 Labs, 385
 Library, 126
 Machine, 57

MISC, 1
MISC 95, 312
Network, 75
OpenLDAP, 385
Paypal, 17
Petazzoni, 263
PostgreSQL, 106, 115
Postgresql, 368
Qui utilise, 17
Registry, 80
registry, 110
Stéphane Beuret, 97
Store, 126
Swarm, 59
swarm, 279
swarm single node:, 61
Sybase, 168
Tutoriels, 175
Videos, 101
volume, 368
volumes, 77
Windows, 265

docker
attach, 63
build, 63
build –no-cache, 63
commands, 63
commit, 63
compose, 252
cp, 63
diff, 63
engine, 21
exec, 63
export, 63
FAQ, 87
help, 63
history, 63
Hub, 387
image, 196
images, 63
inspect, 63, 387
kill, 63
Local dev, 242
login, 63
logs, 63
machine, 262
people, 93
ps, 63
pull, 63
rename, 63
ROS, 311
run, 63, 387
run –detach-keys, 63
search, 63

stop, 63
tag, 63
versions, 23
volume, 63
volumes, 243

Docker Best practices, 52
Docker client, 85
Docker Compose
 Production, 30
Docker daemon, 85
Docker Hub
 hello-world, 387
Docker hub
 Explore, 126
Docker image, 86
Docker Registry
 examples, 84
 Implementations, 82
docker run
 help, 387
Docker Store, 86
Docker Swarm
 Articles, 61
 Articles (2018), 61

docker-ce
 17.06.0-ce (2017-06-23, 02c1d87), 24
 17.12.1-ce (2018-02-27), 24
 18.03.1-ce (2018-04-26, 9ee9f40), 24
 18.06.0-ce (2018-07-18), 24
 18.06.1-ce (2018-08-21), 24
 18.09-ce (2018-11-08), 24

docker-compose
 1.21.2 (2018-05-02), 34
 1.22 (2018-07-18), 34
 build, 26
 commands, 26
 help, 26
 overview, 26
 TIPS, 31
 TIPS (2018), 31
 up, 26
 versions, 33

docker-compose.yml, 85
 2018-06, 106
 Docker, 368
 docker-compose_for_existing_database.yml, 115

docker-compose_for_existing_database.yml
 docker-compose.yml, 115

Dockerfile, 86
 Adam King, 281
 CMD, 215
 COPY, 220
 Docker, 74
 ENTRYPOINT, 215

- Exemple Flask, 399
- files, 220
- Images, 209
- MAINTAINER (deprecated), 74
- Multi-stage builds, 223
- Dockerfiles
 - 2018-06, 106
 - Best practices, 50
 - Tips, 228
- Dockerized cluster
 - swarm, 279
- downloading
 - images, 196
- E**
- engine
 - docker, 21
- ENTRYPOINT
 - Dockerfile, 215
- Essaim, 87
- Examples
 - Compose file, 44
- examples
 - Docker Registry, 84
- exec
 - docker, 63
- Exemple Flask
 - Dockerfile, 399
- Explore
 - Docker hub, 126
- Export
 - pg_dump -U postgres --clean --create -f db.dump.sql
db_id3_intranet, 368
 - PostgreSQL, 119, 368
- export
 - docker, 63
- F**
- FAQ
 - docker, 87
- files
 - copying, 220
 - Dockerfile, 220
- first
 - containers, 188
- Fischer
 - Bret, 95
- G**
- Garrouste
 - Jérémie, 263
- Get Started
 - Part2, 269
 - Part3, 274
- Part4, 279
- Tutoriels, 268
- GitLab
 - Registry, 128
- Gitlab
 - Hébergeurs, 90
 - Images, 170
 - Registry, 82
- gitlab ARM
 - Compose, 47
- GitLab Registry
 - Private, 130
- Glossaire, 84
- Golang
 - Images, 144
- H**
- Hébergeurs
 - Docker, 90
 - Gitlab, 90
- HeidiSQL
 - PostgreSQL, 117
- Hello-world, 175
 - Image, 175
- hello-world
 - Docker Hub, 387
- help
 - docker, 63
 - docker run, 387
 - docker-compose, 26
- History
 - Avril 2018, 184
- history
 - docker, 63
- Howto
 - run a shell in our running container ;, 87
- https
 - Traefik, 106
- Hub
 - docker, 387
- hub
 - Docker, 226
- Hyper-V, 86
- Hyperviseur, 86
 - hyperviseur, 86
- I**
- id3admin
 - USER, 115
- IDs
 - containers, 188, 189
- Image
 - Alpine, 131
 - Apache HTTP Bitnami server, 150

Centos, 136
Debian, 132
Hello-world, 175
MiKTeX, 172
pipenv, 140
rabbitmq, 169
static-site, 175
Ubuntu, 134

image
 class, 196
 docker, 196
 security, 204
 tag, 204
 tags, 196

Images
 Anaconda3, 173
 Apache HTTP server, 148
 Apache Tomcat, 157
 Building, 204
 Docker, 126
 Dockerfile, 209
 Gitlab, 170
 Golang, 144
 MariaDB, 167
 Nginx, 160
 Node, 144
 OpenJDK, 145
 PHP, 142
 PostgreSQL, 161
 Python, 138
 Redmine, 170
 Ruby, 143
 wordpress, 171

images
 docker, 63
 downloading, 196
 namespaces, 196
 Publishing, 226

Implementations
 Docker Registry, 82

Import
 PostgreSQL, 368
 psql -U postgres -f .\db.dump.sql, 368

Inside
 Container, 99, 236

inspect
 docker, 63, 387
 JSON, 231

Installation
 Docker, 15

instance
 container, 196

Intro
 Avril 2018, 177

Introduction
 Docker, 1

ip
 netns, 236

J

Jérémy
 Garrouste, 263

Jérôme
 Docker, 263
 Petazzoni, 263

Jérôme Petazzoni
 2018-03-29, 110
 Avril 2018, 177
 Tutoriaux, 263

Jacob
 Cook, 286

Jacob Cook
 Tutoriel, 286

Janetakis
 Nick, 95

Janvier/Février 2018
 MISC, 1

Jeff Triplett
 Django, 106
 Docker, 106

Joe Jasinski
 Django, 106
 Docker, 106

JSON
 inspect, 231

K

kill
 docker, 63

King
 Adam, 281

Kubernetes
 Stéphane Beuret, 97

L

légère
 Virtualisation, 1

Labels
 Containers, 234

Labs
 Docker, 385
 networking, 75

Lacey
 Williams, 304

Lacey Williams
 2018-05, 107
 2018-06, 106
 Tutoriel, 304

LaTeX, [172](#)

Miktex, [172](#)

letsencrypt

Traefik, [106](#)

Library

Docker, [126](#)

Local dev

docker, [242](#)

login

docker, [63](#)

logs

container, [188](#), [189](#)

docker, [63](#)

M

Machine

Docker, [57](#)

machine

docker, [262](#)

MAINTAINER

deprecated, [74](#)

MAINTAINER (deprecated)

Dockerfile, [74](#)

MariaDB

Images, [167](#)

message-oriented

middleware, [169](#)

Mickael

Bright, [97](#)

Mickael Baron

[2018-05](#), [107](#)

Tutoriaux, [311](#)

middleware

message-oriented, [169](#)

MiKTeX

Image, [172](#)

Miktex

LaTeX, [172](#)

MISC

Docker, [1](#)

Janvier/Février 2018, [1](#)

MISC 95

Docker, [312](#)

Multi-stage

builds, [223](#)

Multi-stage builds

Dockerfile, [223](#)

N

namespace

root, [196](#)

self_hosted, [196](#)

user, [196](#)

namespaces

images, [196](#)

Naming

Containers, [231](#)

netns

ip, [236](#)

Network

Docker, [75](#)

Network drivers

Container, [241](#)

Network model

Container, [241](#)

networking

Labs, [75](#)

Networking basics

Container, [240](#)

News, [103](#)

[2017-08](#), [125](#)

[2018-01](#), [118](#)

[2018-01-29](#), [124](#), [281](#)

[2018-01-30](#), [123](#)

[2018-01-31](#), [119](#)

[2018-02](#), [110](#)

[2018-03](#), [110](#)

[2018-04](#), [109](#)

[2018-05](#), [107](#)

[2018-06](#), [106](#)

[2018-08](#), [105](#)

[2018-09](#), [105](#)

[2018-11](#), [105](#)

[2018-11-08](#), [105](#)

[2017](#), [125](#)

[2018](#), [105](#)

Nginx

Images, [160](#)

Nick

Janetakis, [95](#)

Nick Janetakis, [95](#)

Best practices, [95](#)

Node

Images, [144](#)

non-interactive

container, [188](#)

O

OpenJDK

Images, [145](#)

OpenLDAP

Docker, [385](#)

Orcheteur de conteneurs, [87](#)

Overriding

CMD, [215](#)

Overview

Avril 2018, [177](#)

overview

docker-compose, 26

P

Part2

 Get Started, 269

Part3

 Get Started, 274

Part4

 Get Started, 279

Paypal

 Docker, 17

People

 Bret Fischer, 95

people

 docker, 93

Petazzoni

 Docker, 263

 Jérôme, 263

pg_dump -U postgres --clean --create -f db.dump.sql

 db_id3_intranet

 Export, 368

PHP

 Images, 142

pipenv

 Image, 140

Pipenv avec Docker

 Tutorial, 316

Play with Docker

 Tutorial, 320

PostgreSQL

 Création, 110

 Docker, 106, 115

 Export, 119, 368

 HeidiSQL, 117

 Images, 161

 Import, 368

Postgresql

 Docker, 368

 Tutorial, 368

Private

 GitLab Registry, 130

Production

 Docker Compose, 30

proxy inverse, 87

ps

 docker, 63

psql -U postgres -f ./db.dump.sql

 Import, 368

Publishing

 images, 226

pull

 docker, 63

Python

 Images, 138

Q

Qui utilise
Docker, 17

R

rabbitmq, 169

 AMQP, 169

 celery, 169

 Image, 169

Redmine

 Images, 170

Registry

 Container, 82

 Docker, 80

 GitLab, 128

 Gitlab, 82

registry

 Docker, 110

rename

 docker, 63

restarting

 _container, 194

 containers, 194

reverse proxy, 87

root

 namespace, 196

ROS

 docker, 311

Ruby

 Images, 143

run

 docker, 63, 387

run --detach-keys

 docker, 63

run a shell in our running container ;

 Howto, 87

running

 containers, 188, 189

S

Samples

 Windows 10, 429

search

 docker, 63

Security

 Aquasec, 93

security

 image, 204

self_hosted

 namespace, 196

sequence

 detach, 194

Service discovery

 Container, 241

shell
 container ;, 87

single node:
 swarm, 61

Stéphane
 Beuret, 97
Stéphane Beuret
 Docker, 97
 Kubernetes, 97

static-site
 Image, 175

stop
 docker, 63

Store
 Docker, 126

Swarm, 87
 Application, 417

 Docker, 59

swarm, 87
 Docker, 279
 Dockerized cluster, 279

 single node:, 61

swarm single node:
 Docker, 61

Sybase, 168
 Docker, 168

T

tag
 docker, 63
 image, 204
 version, 196

tags
 image, 196

TIPS
 docker-compose, 31

Tips
 Dockerfiles, 228

TIPS (2018)
 docker-compose, 31

Traefik
 https, 106
 letsencrypt, 106

Tutoriaux
 Jérôme Petazzoni, 263
 Mickael Baron, 311
 William Vincent, 291

Tutorial
 Avril 2018, 177
 Centos7, 321
 Jacob Cook, 286
 Lacey Williams, 304
 Pipenv avec Docker, 316
 Play with Docker, 320

Postgresql, 368

Tutoriels
 Docker, 175
 Get Started, 268

U

Ubuntu, 134
 Image, 134
 Xenial Xerus, 134

up
 docker-compose, 26

USER
 id3admin, 115

user
 namespace, 196

V

version
 tag, 196

versions
 docker, 23
 docker-compose, 33

Videos
 Docker, 101

Vincent
 William, 291

Virtual machine, 87

Virtualisation
 légère, 1

volume
 Docker, 368
 docker, 63

volumes
 Docker, 77
 docker, 243

W

Web
 Apps, 399

William
 Vincent, 291

William Vincent
 Tutoriaux, 291

Williams
 Lacey, 304

Windows
 Docker, 265

Windows 10
 Samples, 429

wordpress
 Images, 171

X

Xenial Xerus

Ubuntu, 134