Table des matières

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EIHS-3DOC** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1
Principes fondamentaux

1. Positionnement de Docker 19
   - 1.1 La problématique racine 19
   - 1.2 Approche par augmentation des ressources physiques 20
   - 1.3 Approche par parallélisation des traitements 21
   - 1.4 Approche par virtualisation 22
   - 1.5 Tentatives de réduction de la consommation de ressources 24
   - 1.6 Comment Docker règle radicalement le problème 25
   - 1.7 Positionnement de Docker par rapport à la virtualisation 28
   - 1.8 Positionnement de Docker par rapport aux clusters 30
   - 1.9 Positionnement de Docker par rapport aux approches de bacs-à-sable 32
2. Principe des conteneurs 36
   - 2.1 Les apports de Docker 36
   - 2.2 Principe des conteneurs industriels 37
   - 2.3 Docker et l'approche normalisée 38
3. Les fondements de Docker 40
   - 3.1 Les technologies Linux clés pour Docker 40
     - 3.1.1 Namespaces 40
     - 3.1.2 Mise en œuvre d'un namespace 41
     - 3.1.3 cgroups 43
     - 3.1.4 chroot 43
   - 3.2 Autres dépendances du système 45
     - 3.2.1 netfilter et iptables 45
     - 3.2.2 capabilities 46

3.2.3 AppArmor et SELinux 46
3.3 Architecture du moteur Docker 46
3.3.1 LXC 46
3.3.2 libcontainer 47
3.3.3 containerd 48
3.4 Architectures complémentaires 50
3.4.1 rkt 50
3.4.2 Container Runtime Interface 50
3.4.3 Un foisonnement d'alternatives 51
3.5 Système de fichiers en couches 51
3.5.1 Principe d'isolation des fichiers 51
3.5.2 Approche par virtualisation 52
3.5.3 Utilité des systèmes de fichiers en couches 53
3.5.4 Gestion des modifications de fichiers 54
3.5.5 Dernière couche en écriture 55
3.5.6 Technologies utilisées 57
4. Les plus de Docker 57
4.1 Au-delà du cloisonnement Linux 57
4.2 L'approche un conteneur = un processus 58
4.3 L'écosystème Docker 59
5. Architectures de services 61
5.1 Historique des architectures de services 62
5.1.1 Principes 62
5.1.2 Approche EAI 62
5.1.3 Approche SOA 62
5.1.4 Microservices 63
5.1.5 Lien à l'urbanisation des SI 64
5.2 Architecture de microservices 64
5.2.1 Principe 64
5.2.2 Avantages 67
5.2.3 Inconvénients 69
5.3 Apport de Docker 70
5.4 Fil conducteur 71
6. Des conteneurs sans Docker avec Podman 71

Chapitre 2
Installation

1. Éditions et canaux 73
   1.1 Un standard sous Linux 73
   1.2 Les canaux 74
   1.3 Les éditions 74
2. Utiliser des machines dans le cloud 75
   2.1 Amazon AWS 75
      2.1.1 Offre 75
      2.1.2 Création de la machine virtuelle 78
      2.1.3 Accès à la machine virtuelle 85
      2.1.4 Gestion de la consommation 89
      2.1.5 Suppression de la machine virtuelle 90
   2.2 Microsoft Azure 92
      2.2.1 Offre 92
      2.2.2 Création de la machine virtuelle 93
      2.2.3 Accès à la machine virtuelle 99
      2.2.4 Suppression de la machine virtuelle 108
   2.3 Google Cloud Platform 108
      2.3.1 Offre 108
      2.3.2 Création de la machine virtuelle 111
      2.3.3 Accès à la machine virtuelle 117
      2.3.4 Suppression de la machine virtuelle 118
3. Installation de Docker 120
   3.1 Installation de Docker sur Linux 120
      3.1.1 Prérequis 120
      3.1.2 Gestionnaires de paquets 121
      3.1.3 Installation par script 121
      3.1.4 Cas particulier de l'installation sur Clear Linux 122
   3.2 Le paradoxe Docker sous Windows 125
      3.2.1 Des technologies différentes 125
      3.2.2 Docker et Hyper-V 127
      3.2.3 L'âge de la maturité 127

3.3 L'outil Docker pour Windows 128
3.3.1 Outil d'installation 128
3.3.2 Choix du canal 131
3.3.3 Assistant d'installation 132
3.3.4 Mises à jour 135
3.3.5 Support natif de conteneurs Linux sous Windows avec WSL 2 136
3.3.6 Les nouveautés de Docker Desktop 144
3.3.7 Réduction de la taille du disque virtuel 147
3.3.8 Désinstallation 148
3.4 Docker sous Windows sans Docker Desktop 149
3.4.1 Installation de Docker 149
3.4.2 Ligne de commande docker depuis Windows 150
3.4.3 Ouverture du démon Docker sur TCP (distribution avec systemd) 152
3.4.4 Variante pour les distributions sans systemd 152
3.4.5 Débogage réseau sous WSL 153
3.4.6 Accès à Docker et WSL 154
3.5 Docker pour Windows Server 155
3.5.1 Prérequis pour Windows Server 2022 155
3.5.2 Installation 156
3.5.3 Utilisation de machines préinstallées 161
3.5.4 Exécution de conteneurs Linux 162
3.5.5 Paramétrage système 166
3.6 Utilisation de Docker avec Vagrant 167
3.6.1 Principe 167
3.6.2 Création d'une machine 168
3.6.3 Provisionnement de Docker 173
3.6.4 Aménagement de l'accès à la machine 173
3.6.5 Connexion à la machine provisionnée 176
3.6.6 Poursuite des opérations Vagrant 179

Chapitre 3
Premiers pas avec Docker

1. Hello World, Docker 181
   1.1 Démarrage d'un conteneur simple 181
   1.2 Détails des opérations effectuées 182
      1.2.1 Récupération d'une image 182
      1.2.2 Identité de l'image 184
      1.2.3 Taille des images 187
      1.2.4 Anatomie de l'image Linux 189
      1.2.5 Anatomie de l'image Windows 194
      1.2.6 Lancement du processus 195
      1.2.7 Exécution du processus dans un conteneur 196
      1.2.8 Opérations suivant l'exécution 197
2. Utiliser des images Docker préexistantes 200
   2.1 Le registre Docker Hub 200
      2.1.1 Le principe 200
      2.1.2 Recherche et qualification d'images 201
      2.1.3 Exemple de recherche 205
      2.1.4 Cas des images communautaires 209
      2.1.5 Compléments sur les images officielles 212
      2.1.6 Recherche par la ligne de commande 214
      2.1.7 Précautions sur une image non officielle 217
   2.2 Gestion du compte Docker Hub et dépôts privés 224
      2.2.1 Création d'un compte 225
      2.2.2 Caractéristiques du compte 225
      2.2.3 Automated build 227
      2.2.4 Connexion au compte en ligne de commande 228
      2.2.5 Webhook sur événement de push dans Docker Hub 232
3. Un second conteneur 234
   3.1 Récupération de l'image 234
   3.2 Explication des tags 235
   3.3 Premier lancement 236
   3.4 Lancement en mode interactif 238
   3.5 Persistance des modifications sous forme d'une image 240

3.6 Prise en main du client Docker 243
3.6.1 Ménage dans les conteneurs 243
3.6.2 Ménage dans les images 244
3.6.3 Le grand ménage 244
3.6.4 Suppression automatique à la sortie 245
3.6.5 Affectation d'un nom de conteneur 245
3.6.6 Modification du point d'entrée par défaut 246
3.6.7 Envoi de variables d'environnement 247
3.6.8 Modification du hostname 249
3.7 Manipulation des conteneurs 250
3.7.1 Lancement en mode bloquant 250
3.7.2 Lancement en arrière-plan 251
3.7.3 Gestion correcte du cycle de vie des conteneurs 256
3.7.4 Exposition de fichiers 258
3.7.5 Supervision des conteneurs 260
4. Retours sur les premiers pas 264

Chapitre 4

Création et gestion d'images Docker

1. Création manuelle d'une nouvelle image 267
1.1 Objectif 267
1.2 Approche 267
1.3 Difficultés 268
1.4 Conclusion 268
2. Utilisation d'un fichier Dockerfile 269
2.1 Intérêt des fichiers Dockerfile 269
2.2 Utilisation d'un fichier Dockerfile 272
2.3 Anatomie d'un fichier Dockerfile 275
2.3.1 FROM 275
2.3.2 RUN 276
2.3.3 ENV 276
2.3.4 VOLUME 278
2.3.5 COPY 285
2.3.6 ENTRYPOINT 285
2.3.7 EXPOSE 287

2.3.8 CMD 287
2.4 Notre premier fichier Dockerfile 288
2.4.1 Fonctionnalité souhaitée 288
2.4.2 Création et test du script 288
2.4.3 Création du Dockerfile 290
2.4.4 Génération de l'image 292
2.4.5 Lancement du conteneur 293
2.4.6 Arrêt et relance du conteneur 295
2.4.7 Une méthode moins brutale 295
2.4.8 Gestion des paramètres 296
2.4.9 Reconstruction de l'image et cache 298
2.5 Commandes additionnelles 300
2.5.1 Gestion des fichiers 300
2.5.2 Notion de contexte 301
2.5.3 Retours sur l'affectation du processus à démarrer 301
2.5.4 Remarque sur le format ligne de commande ou exécution . . . 304
2.5.5 Outils de compilation 306
2.5.6 Commandes diverses 307
3. Partage et réutilisation simple des images 310
3.1 Envoi sur votre compte Docker Hub 310
3.2 Export et import sous forme de fichiers 316
4. Bonnes pratiques 317
4.1 Principe du cache local d'images 317
4.2 Principe du cache à la compilation 322
4.2.1 Retour sur les images intermédiaires 322
4.2.2 Anatomie d'une compilation d'image 326
4.2.3 Analyse d'une modification du fichier Dockerfile 328
4.2.4 Gestion correcte des étiquettes 331
4.2.5 Invalidation du cache par modification de l'image de base . . . 334
4.2.6 Invalidation du cache par modification du contexte 339
4.3 Conséquences sur l'écriture des fichiers Dockerfile 341
4.3.1 Le problème sur les opérations non idempotentes 341
4.3.2 Contournement du problème de cache 344
4.3.3 Effets bénéfiques sur le nombre et la taille des images 347
4.3.4 Ordonnancement des commandes dans le fichier Dockerfile . 349

4.4 Ajout d'une image de cache intermédiaire . . . 349
4.5 Mise en œuvre d'un cache de paquetages . . . 351
4.6 Conséquences sur le choix des images de base . . . 354
4.6.1 La bonne image de base . . . 354
4.6.2 Votre propre image de base . . . 356
4.6.3 Adapter l'image de base avec les compilations multi-stage . . . 359
4.7 Arborescence recommandée . . . 362
4.7.1 Avantage d'une arborescence type . . . 362
4.7.2 Intégration des fichiers . . . 363
4.7.3 Limitation du contexte . . . 365
4.8 La question du processus unique . . . 367
4.8.1 Principe général . . . 367
4.8.2 Exception au principe général avec Supervisor . . . 367
4.8.3 Critique . . . 369
4.8.4 Approche intermédiaire . . . 370
5. Mise en œuvre d'un registre privé . . . 371
5.1 Objectifs . . . 371
5.2 Utilisation d'un service cloud déjà préparé . . . 372
5.2.1 Azure Container Registry . . . 372
5.2.2 Fonctionnalités de suivi . . . 378
5.2.3 Fonctionnalités de sécurité . . . 382
5.2.4 Fonctionnalités d'automatisation . . . 384
5.2.5 Notes finales . . . 385
5.3 Approches complémentaires . . . 387
5.3.1 L'API du registre . . . 387
5.3.2 Mise en place d'un miroir . . . 388
6. Incorporation dans le cycle de développement . . . 391
6.1 Positionnement de Docker dans une usine logicielle . . . 391
6.1.1 Déploiement . . . 393
6.1.2 Compilation . . . 394
6.1.3 Infrastructure de l'usine logicielle . . . 395
6.1.4 Gestion des tests . . . 397
6.1.5 Utilisation pour les machines supports de tests . . . 398
6.1.6 Registre pour l'ALM . . . 399

6.2 Exemple de mise en œuvre 400
6.2.1 Choix des outils 400
6.2.2 Création du projet 401
6.2.3 Analyse du fichier Dockerfile généré 406
6.2.4 Prérequis pour l'intégration continue 409
7. Encore plus loin avec Docker 410
7.1 Applications en circuit fermé 411
7.2 Base de données SQLServer 412
7.3 Tests unitaires avec TestContainers 414
7.4 Développement avec Aspire 416
7.5 Déploiement d'outils d'intelligence artificielle 417
7.6 Docker pour des applications Windows 418
7.7 Images natives/distroless 419
7.8 Docker comme une commodité 420

Chapitre 5
Mise en œuvre pratique

1. Présentation de l'application exemple 423
1.1 Architecture 423
1.2 Création d'un fichier des mots de passe 428
1.2.1 Principe de sécurité 428
1.2.2 Installation de KeePass 428
1.2.3 Création du fichier de sécurité 429
1.2.4 Remplissage du fichier de mots de passe 434
1.3 Mise en place des certificats de sécurité 436
1.3.1 Certificat racine 437
1.3.2 Certificat client 441
1.4 Prérequis 445
1.5 Paramétrage 445
1.5.1 Paramétrage de l'IAM 446
1.5.2 Paramétrage de RecepteurMessages 455
1.5.3 Paramétrage du projet API 458
1.5.4 Autres projets 459
1.5.5 Paramétrage de la solution 461

1.6 Première utilisation 463
1.6.1 Scénario nominal de test 463
1.6.2 Persistance 466
1.6.3 Observation des documents 467
1.7 Utilité 469
1.8 Principes à l'œuvre 469
1.8.1 Un mot sur les architectures de services 469
1.8.2 Lien avec la programmation SOLID 470
2. Adaptation à Docker de l'application exemple 471
2.1 Préparation de l'environnement 471
2.2 Principes de construction 471
2.3 Détails du service RecepteurMessages 472
2.3.1 Fonctionnement 473
2.3.2 Construction du fichier Dockerfile de base 475
2.3.3 Classification du support des paramètres 479
2.3.4 Cas particulier des paramètres de sécurité 481
2.3.5 Modification du projet pour gestion des paramètres 482
2.3.6 Évolution du fichier Dockerfile 487
2.4 Lancement de RecepteurMessages en mode Docker 491
2.4.1 Compilation 491
2.4.2 Test de lancement 493
2.4.3 Conteneurs en réseau 495
2.4.4 Utilisation d'un volume pour pointer sur le certificat 497
2.5 Adaptation de Visual Studio 498
2.5.1 Paramétrage du projet en mode Docker 498
2.5.2 Tests d'intégration 503
2.5.3 Publication intégrée 505
2.6 Retour sur le problème de secret 506
2.6.1 Approche simpliste par variable d'environnement 507
2.6.2 Gestion des secrets dans Docker 508
2.6.3 Mise en place du fichier pour Docker Compose 510
2.6.4 Mise en place de la trace de diagnostic 512
2.6.5 Description du problème d'ordonnancement 514
2.6.6 Validation du fonctionnement en mode Docker 516
2.7 Problème des paramètres liés à XKCD 517

2.8 Traitement du projet API. 519
2.8.1 Passage en mode Docker du projet 519
2.8.2 Classification des paramétrages disponibles 520
2.8.3 Ajustement du fichier Dockerfile 522
2.8.4 Ajout des valeurs pour les paramètres 522
2.8.5 Mise à jour du fichier Docker Compose. 523
2.8.6 Correction paramétrage d'appel du service API 524
2.8.7 Test du projet en mode Docker 526
2.9 Traitement du projet Server. 526
2.9.1 Passage en mode Docker du projet 527
2.9.2 Classification des paramètres disponibles 528
2.9.3 Ajustement du fichier Dockerfile 528
2.9.4 Ajout des valeurs pour les paramètres 529
2.9.5 Mise à jour du fichier Docker Compose. 530
2.10 Cas particulier des paramètres du projet Client 531
2.11 Test de l'application passée en Docker 531
2.11.1 Principe. 531
2.11.2 Débogage du problème de volume 532
2.11.3 Débogage du problème de nom de serveur RabbitMQ 538
2.11.4 Débogage du problème d'accès à l'IAM 541
2.11.5 Débogage du chargement des personnes 546
2.11.6 Débogage de l'authentification serveur 549
2.12 Cas particulier de la GED 553
2.13 Récupération du livrable dans le second stage 554
2.13.1 Installation de Tomcat sur la JRE. 554
2.13.2 Gestion des types CMIS 555
2.13.3 Intégration du entrypoint. 555
2.13.4 Analyse du script entrypoint 556
2.14 État atteint 557

Chapitre 6
Orchestration par Docker Compose

1. Redéployer automatiquement avec Docker Compose 559
1.1 Principe de Docker Compose 559
1.2 Écriture du fichier compose.yaml 561
1.2.1 Remarques préliminaires 561
1.2.2 Préparation du squelette du fichier 562
1.2.3 Mise en place des dépendances externes 562
1.2.4 Ajout de la détection de statut des services 565
1.2.5 Inclusion du projet RecepteurMessages 568
1.2.6 Ajout du projet API 571
1.2.7 Finalisation avec le projet Server 572
1.3 Mise en œuvre 574
1.3.1 Exclusion du mode HTTPS 574
1.3.2 Premier lancement 574
1.3.3 Retour sur le problème d'authentification 576
1.3.4 Tentative de résolution par le réseau hôte 579
1.3.5 Stratégie d'externalisation de l'IAM 581
1.4 Mise en œuvre avec IAM externalisée 582
1.4.1 Préparation d'une machine dans le cloud 582
1.4.2 Ajout de Docker dans Clear Linux 586
1.4.3 Préparation de l'application Keycloak complète 588
1.4.4 Passage de Keycloak en HTTPS 590
1.4.5 Ajout de la persistance pour la configuration Keycloak 594
1.4.6 Initialisation de la base de données 599
1.4.7 Gestion avec un service Linux 600
1.4.8 Réduction de la surface d'attaque 602
1.4.9 Accès à l'interface de gestion PostgreSQL 602
1.4.10 Ajustement de l'application exemple 605
1.4.11 Test final 606

1.5 Débogages complémentaires du mode Docker Compose 609
1.5.1 Astuce préliminaire 609
1.5.2 Gestion de la génération de PDF en mode Linux 610
1.5.3 Intégration des polices pour la génération de PDF 610
1.5.4 Exposition différenciée des ports de la GED 611
1.5.5 Exposition en HTTPS du projet API 613
1.5.6 Passage de l'appel d'API en HTTPS 614
1.5.7 Ménage dans les paramètres 615
1.5.8 Utilisation de profils 619
1.5.9 Réutilisation de fichiers Docker Compose 620
2. Fonctionnalités supplémentaires de Docker Compose 622
2.1 Retour sur la gestion des conteneurs 622
2.2 Parallélisation des traitements 623
2.3 Autre avantage de l'approche par conteneurs 625
2.4 Limites de l'application exemple 627
2.5 Intégration dans Visual Studio 628
2.5.1 Argumentaire 628
2.5.2 Mise en œuvre 630
2.5.3 Gestion des profils de lancement 632
3. Exploitation d'une infrastructure Docker 634
3.1 Le réseau dans Docker 634
3.1.1 Mode de fonctionnement standard (bridge) 634
3.1.2 Modes de fonctionnement alternatifs 637
3.1.3 Support des liens entre conteneurs 639
3.1.4 Gestion des réseaux overlay 641
3.1.5 Alias préinstallés 645
3.2 Les volumes Docker 646
3.2.1 Le problème de la persistance 646
3.2.2 Les volumes comme solution simple 646
3.2.3 Lien direct sur un répertoire local 648
3.2.4 Partage de volumes 649
3.2.5 Gestion des volumes orphelins 650
3.2.6 Sophistication de l'approche 651
3.2.7 Application à la gestion des logs 651
3.2.8 Volumes déclarés 653
3.2.9 Plugins pour la gestion avancée des volumes 653

4. Exemples de fichier Dockerfile pour d'autres plateformes 654
4.1 Java. 655
4.1.1 Image Java 655
4.1.2 Image openjdk 656
4.1.3 Image Maven 656
4.2 PHP. 659
4.2.1 Mode ligne de commande 659
4.2.2 Mode serveur web 660
4.3 Node.js. 661
4.4 Go. 662
4.5 Python 662
4.6 Un mot sur les DevContainers. 663

Chapitre 7
Déploiement en cluster par une usine logicielle

1. Le besoin d'orchestration. 665
1.1 Objectif 665
1.2 Approche théorique 668
1.2.1 La problématique de montée en charge 668
1.2.2 La solution découplée. 670
1.2.3 Conséquences sur l'approche initiale 671
1.3 Lien aux microservices 672
1.3.1 Orchestration des services. 672
1.3.2 Élasticité 673
1.3.3 Robustesse 673
1.4 Fonctionnement pratique. 674
1.4.1 Notion de réseau 674
1.4.2 Les différents types de nœuds. 675
1.4.3 Fonctionnalités du cluster. 676
2. L'approche Docker Swarm. 676
2.1 Gestion du cluster Swarm 677
2.1.1 Prérequis 677
2.1.2 Alternative aux prérequis 678
2.1.3 Initialisation. 682
2.1.4 Liaison des agents. 684

2.1.5 Ajout d'un manager 687
2.1.6 Les limites à l'ajout de managers 689
2.1.7 Promotion d'un nœud 691
2.1.8 Suppression d'un nœud 692
2.2 Test du cluster avec un cas simple 695
2.3 Déploiement manuel sur le cluster Swarm 699
2.3.1 Application exemple 699
2.3.2 Préparation des images 700
2.3.3 Lancement des services 701
2.3.4 Premier test 706
2.3.5 Mise en place d'un réseau overlay dédié 708
2.3.6 Validation du fonctionnement 709
2.3.7 Passage à l'échelle 711
2.3.8 Suppression des ressources déployées 713
2.4 Déploiement de l'application microservices 714
2.4.1 Récupérer l'application exemple 714
2.4.2 Le retour partiel de Docker Compose 714
2.4.3 Détail du vocabulaire 715
2.4.4 Compilation éventuelle des images 717
2.4.5 Lancement par Docker Compose 718
2.4.6 Déploiement d'une stack 719
2.4.7 Autres avertissements possibles 722
2.4.8 Diagnostic et validation du fonctionnement 724
2.4.9 Mise au point du déploiement 730
2.4.10 Utilisation de l'ingress 736
2.4.11 Déploiement de l'IAM sur Docker Swarm 738
2.4.12 Ajustement de la stack applicative 741
2.4.13 Débogage du service de traitement des messages 743
2.5 Considérations additionnelles 748
2.5.1 Provenance des images 748
2.5.2 Passage à l'échelle 748
2.5.3 Mise à jour des services avec de nouvelles images 749
2.5.4 Gestion des contraintes 753
2.5.5 Arrêt de la stack 754

3. Outils avancés d'exposition 756
3.1 Traefik 756
3.1.1 Utilité 756
3.1.2 Lancement 756
3.1.3 Mise en œuvre sur l'application exemple 758
3.1.4 Vérification du fonctionnement 760
3.2 Caddy 763
3.2.1 Présentation 763
3.2.2 Réunification des stacks 763
3.2.3 Remplacement de Traefik par Caddy 764
3.2.4 Exposition simpliste pour test 765
3.2.5 Ajout d'un fichier de configuration 768
3.2.6 Modifications complémentaires 769
3.2.7 Paramétrage de Keycloak pour le proxy 770
3.2.8 Conclusion sur Caddy 772
4. Compléments sur la variabilisation 774
4.1 Utilité des variables 774
4.2 Mise en œuvre dans Docker 775
4.3 Application à la solution exemple 777
5. Introduction à Kubernetes 780
5.1 Positionnement 780
5.2 Comparaison avec Docker Swarm 781
5.3 Concepts 782
5.3.1 Pods 782
5.3.2 Namespaces 783
5.3.3 Services 783
5.3.4 IngressController 783
5.3.5 Volumes 784
5.3.6 ConfigMap 785
5.3.7 Deployment 786
5.4 Conseils pour le déploiement 787
5.4.1 Provisionnement d'un cluster Kubernetes 787
5.4.2 Installation d'un client Kubernetes 787
5.4.3 Spécificités de déploiement de l'application exemple 788

6. Intégration et déploiement continus 789
6.1 Approche 789
6.2 Intégration continue de l'application exemple 790
6.2.1 Création du projet Azure DevOps 790
6.2.2 Création d'un pipeline de compilation pour le projet Server . 791
6.2.3 Correction du problème de compilation 801
6.2.4 Remarques sur l'intégration des plateformes 805
6.3 Déploiement continu avec Azure DevOps 806
6.3.1 Objectif 806
6.3.2 Mécanisme utilisé 806
6.3.3 Paramétrage dans Azure DevOps 808
6.3.4 Ménage 814
6.4 Alternative de mise à jour continue 814
6.5 Fonctionnalités supplémentaires d'usine logicielle 815
6.5.1 Scan des images 815
6.5.2 Alternative Docker Scout 818
6.5.3 Software Bill Of Material 820
7. Azure Container Instances 821
7.1 Principe 821
7.2 Préparation d'une image 821
7.3 Lancement du conteneur 822
7.4 Correction de l'erreur et relance 825
7.5 Déploiement d'un groupe 827
7.6 Coût et effort 829

Conclusion 833

Index 835

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence de l'ouvrage **EI3KUB** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

1. Présentation de Kubernetes 43
1.1 Un peu d'histoire 43
1.2 Qu'est-ce qu'un conteneur ? 44
1.3 Les conteneurs avant Docker 44
1.4 Pourquoi utiliser des conteneurs ? 45
1.5 Problèmes introduits avec les conteneurs 46
1.6 À quoi va servir Kubernetes ? 46
1.7 Ressources externes 47
2. Un mot sur l'application 47
2.1 Rien ne sert de courir 47
2.2 Les douze facteurs applicatifs 48
2.3 Microservices vs Monolithes 48

Chapitre 1
Introduction

1. Cibles et objectifs de l'ouvrage 49
2. Prérequis techniques et ressources documentaires 50
2.1 Prérequis techniques 50
2.2 Ressources documentaires 50
2.3 Récupération des fichiers d'exemples 51
3. Présentation générale 51
3.1 Prérequis 51
3.2 Utilisation de Kubernetes 52
3.3 Installation et configuration de Kubernetes 52
3.4 Extension du cluster Kubernetes et notions avancées 53

3.5 Déploiement et intégration continue 53
3.6 Conventions utilisées 54

Chapitre 2
Installation de l'environnement Kubernetes

1. Objectifs du chapitre et prérequis 55
2. Alternative à l'installation en local 56
   2.1 Pourquoi ces alternatives ? 56
   2.2 Utilisation d'un service managé 56
   2.3 Service Killercoda 57
3. Mise en place de la commande kubectl 58
   3.1 À quoi sert kubectl ? 58
   3.2 Installation de kubectl 58
      3.2.1 Installation sous Debian/Ubuntu 58
      3.2.2 Installation sous CentOS/RHEL ou Fedora 59
      3.2.3 Installation à l'aide d'Arkade 59
   3.3 Vérification de l'installation 62
   3.4 Configuration de l'autocomplétion 62
      3.4.1 Présentation du mécanisme d'autocomplétion 62
      3.4.2 Fichier profile à modifier 62
      3.4.3 Autocomplétion sur kubectl 63
      3.4.4 Utilisation de la variable SHELL 63
      3.4.5 Mise en place d'un alias 64
4. Mise en place de Minikube 64
   4.1 Pourquoi faire appel à Minikube ? 64
   4.2 Téléchargement et installation de Minikube 65
   4.3 Vérification de l'installation de Minikube 65
   4.4 Mise en place de l'autocomplétion 66

5. Installation du cluster Kubernetes avec Minikube 66
5.1 Options de lancement 66
5.2 Installation de Docker 67
5.2.1 Installation de Docker Community Edition sur Ubuntu 67
5.2.2 Installation alternative 68
5.2.3 Configuration des accès à Docker 68
5.2.4 Vérification de l'installation de Docker 68
5.3 Installation de l'hyperviseur VirtualBox 69
5.4 Installation de l'hyperviseur KVM/libvirt 70
5.5 Configuration de l'utilisateur courant 70
5.6 Déploiement du cluster avec Minikube 71
5.6.1 Création de la machine Minikube 71
5.6.2 Arrêt/démarrage de la machine Minikube 75
5.6.3 Choix du conteneur runtime 76
5.6.4 Extensions de Minikube 76
5.6.5 Suppression de la machine Minikube 78
6. Quelques notions sur le format YAML 79
6.1 Déclaration de couples clés/valeurs 79
6.2 Les tableaux en YAML 80
6.3 Les structures clé/valeur ou table de hachage 81
6.4 Tableau de table de hachage 83

Chapitre 3
Tableau de bord et ligne de commande

1. Objectifs du chapitre et prérequis 85
2. Préambule 86
2.1 Origine du nom et du logo 86
2.2 Pourquoi utiliser Kubernetes ? 86
2.3 Origine de Kubernetes 87
2.4 Fondation CNCF 87
2.5 Les orchestrateurs du marché 88

3. Le tableau de bord de Kubernetes (dashboard) 89
3.1 Présentation 89
3.2 Tableau de bord Kubernetes sur service managé 89
3.3 Déploiement du dashboard sur Minikube 89
3.4 Accès au dashboard sur Minikube 90
3.5 Structure du tableau de bord 91
3.6 Création d'un déploiement 92
3.6.1 Un petit mot sur Mailpit 92
3.6.2 Lancement du déploiement 93
3.7 État d'un déploiement 95
3.7.1 Consultation de l'état du déploiement 95
3.7.2 Consultation du gestionnaire de réplicas 96
3.7.3 Consultation de l'état d'un pod 98
3.7.4 Journal d'activité du conteneur 99
3.7.5 Scalabilité 99
3.7.6 Mise à jour de l'application 101
3.7.7 Pour résumer 103
4. Présentation de l'outil kubectl 103
4.1 Préambule 103
4.2 Consultation des éléments 103
4.3 Liste des pods 105
4.4 Liste des machines d'un cluster 106
4.4.1 Connexion à la machine Minikube 106
4.4.2 Liste des nœuds d'un cluster 106
4.4.3 Affichage des caractéristiques étendues 107
5. Le moteur Containerd de Minikube 108
5.1 Initialisation de l'environnement 108
5.2 Les conteneurs associés aux pods 109

Chapitre 4
Automatisation et publication d'une application

1. Objectifs du chapitre et prérequis 111
2. Gestion par kubectl d'une application 111
   2.1 Suppression d'un déploiement 111
   2.2 Création d'un déploiement 112
   2.3 État du déploiement 113
   2.4 Mécanisme des réplicas 115
      2.4.1 Consultation des réplicas 115
      2.4.2 Description des réplicas 116
   2.5 État du pod 117
      2.5.1 Liste des pods 117
      2.5.2 Détails de l'état d'un pod 118
   2.6 Accès aux logs des conteneurs 119
   2.7 Accès à l'application Mailpit 120
3. Exposition de services 121
   3.1 Pourquoi utiliser un service ? 121
   3.2 Exposition d'un déploiement via un service 122
   3.3 Vérification du service mailpit 122
   3.4 Que faire en cas d'absence de shell ? 124
      3.4.1 Contexte 124
      3.4.2 Utilisation d'un pod éphémère 125
      3.4.3 Lancement d'un pod de test 126
   3.5 Résilience et scalabilité 127
      3.5.1 Origine du besoin 127
      3.5.2 Scalabilité manuelle 128
      3.5.3 Nombre de pods associés à un déploiement 128
      3.5.4 Arrêter temporairement une application 128

4. Automatisation de déploiement par fichier YAML 130
4.1 Mécanisme de création et mise à jour 130
4.2 Structure YAML d'un déploiement 130
4.2.1 Quelques rappels 130
4.2.2 Récupération d'une structure au format YAML 131
4.2.3 Édition d'un déploiement 132
4.2.4 Squelette pour un déploiement 133
4.2.5 Création d'un déploiement à l'aide d'un fichier 135
4.2.6 Suppression des éléments d'un fichier 136
4.2.7 Gestion de l'idempotence et de la réentrance 136
4.3 Création du service 138
4.3.1 Définition du service 138
4.3.2 Application de la définition du service 139
4.3.3 Gestion de la réentrance 139
4.4 Mécanisme de sélecteur et labels 140
4.5 Regroupement de la création des éléments 142
4.5.1 Création d'un groupe d'objets 142
4.5.2 Consultation de l'état d'un groupe d'objets 143
4.6 Structure des objets 143
4.6.1 Interrogation de Kubernetes avec kubectl 143
4.6.2 Référence de l'API en ligne 144

5. Ingress et reverse proxy 145
5.1 Origine du besoin 145
5.2 Rôle d'un proxy inverse 145
5.3 Activation du contrôleur Ingress dans Minikube 146
5.4 Déclaration d'une règle Ingress 147
5.5 Consultation des règles Ingress 148
5.6 Hôte virtuel et nip.io 150
5.6.1 Hôte virtuel par défaut 150
5.6.2 Présentation du mécanisme de nip.io 150
5.6.3 Configuration du serveur DNS 151
5.6.4 Création d'un hôte virtuel pour Mailpit 153

Chapitre 5
Cycle de vie d'un conteneur dans Kubernetes

1. Objectifs du chapitre et prérequis 157
2. Gestion des crashs d'application 158
   2.1 Consultation de l'état des pods 158
   2.2 Connexion au pod 158
   2.3 Conteneur associé à Mailpit 159
   2.4 Comportement en cas de crash 161
   2.5 État du conteneur après redémarrage du pod 162
   2.6 Container vu depuis Containerd (Minikube) 162
   2.7 Attention au nettoyage 164
3. État d'un conteneur 164
   3.1 Pourquoi scruter l'état d'un conteneur ? 164
   3.2 Readiness vs Liveness 165
   3.3 Utilisation et bonne pratique 166
   3.4 Structure des champs de surveillance 166
   3.5 Vérification de la présence d'un port 168
      3.5.1 Définition de la surveillance 168
      3.5.2 Test d'indisponibilité sur un pod non prêt 169
      3.5.3 État des pods en cas d'indisponibilité 170
      3.5.4 Test d'indisponibilité sur un pod en mauvaise santé . . . 170
      3.5.5 État des pods en cas de problème sur un pod 171
      3.5.6 Attention à la consistance des tests 171
      3.5.7 Uniformisation des tests 173
   3.6 Surveillance HTTP 174
      3.6.1 Pourquoi privilégier ce type de surveillance ? 174
      3.6.2 Surveillance de l'application Mailpit 174
   3.7 Point d'entrée de surveillance HTTP d'une application 175
      3.7.1 Un mot sur les frameworks modernes 175
      3.7.2 Présentation de l'application Flask 175
      3.7.3 Exemple de déclaration 176
      3.7.4 Déploiement de l'application Flask 177

3.7.5 Consultation de l'état de l'application 177
3.8 Lancement d'un shell 179
3.8.1 Principe de fonctionnement 179
3.8.2 Exemple de surveillance d'une base Postgres 179
3.8.3 Déclaration de la commande 180
4. Définition de la capacité d'un pod 180
4.1 Pourquoi définir une capacité ? 180
4.2 Réservation et surallocation 181
4.3 Allocation de ressources à un conteneur 181
4.4 Allocation de ressources à l'application Mailpit 182
4.5 Comportement en cas de saturation des ressources 183
4.5.1 Demande trop importante de CPU 183
4.5.2 Dépassement de la mémoire allouée 184
4.6 Priorité d'un pod 185
4.6.1 Présentation du mécanisme 185
4.6.2 Consultation des types par défaut 186
4.6.3 Consultation des priorités des pods 186
4.6.4 Création d'une classe de priorité 188
4.6.5 Affectation d'une classe de priorité personnalisée 189
4.6.6 Remarque sur les classes de priorité par défaut 190

Chapitre 6
Persistance des données

1. Objectifs du chapitre et prérequis 191
2. Persistance des données 191
2.1 Origine du besoin 191
2.2 Utilisation d'un volume persistant externe 192
2.3 Volumes persistants 193
2.3.1 Structure du volume persistant 193
2.3.2 Création du volume persistant 194

2.4 Persistance de données avec Mailpit 194
2.4.1 Opérations à réaliser 194
2.4.2 Déclaration de l'objet PersistentVolumeClaim 195
2.4.3 État des objets de volume persistant 196
2.4.4 État de la demande de volume persistant 197
2.4.5 Déclaration du point de montage 197
2.4.6 Ajout d'un point de montage sur le conteneur 198
2.4.7 Options de lancement de Mailpit 198
2.4.8 Déclaration entière suite aux modifications 199
2.5 Test de la persistance 201
2.5.1 Utilisation de Mailpit pour l'envoi d'e-mail 201
2.5.2 Ouverture de la communication avec le port SMTP . . . 202
2.5.3 Envoi d'un mail 202
2.5.4 Réduction des droits d'exécution et initialisation 203
2.5.5 Gestion des droits du répertoire de persistance 205
2.5.6 Consultation de l'interface de Mailpit 208
2.5.7 Suppression des pods 209
2.5.8 Vérification du fonctionnement de la persistance 210
3. Classes de stockage 210
3.1 Origine du besoin 210
3.2 Liste des classes de stockage 211
3.3 Détail d'une classe de stockage 212
3.4 Classe de stockage par défaut 212
3.5 Les différentes classes de stockage 213
3.5.1 Les différentes familles 213
3.5.2 Origine de ces familles 213
3.6 Caractéristiques des classes de stockage 214
3.6.1 Modes d'accès 214
3.6.2 Caractéristiques de certains pilotes 215
3.6.3 Liste des pilotes chargés 216
3.7 Déclaration d'une classe de stockage 217
3.7.1 Structure de la déclaration 217
3.7.2 Exemple de déclaration 218

3.8 Test de création automatique d'un volume persistant 219

Chapitre 7
Hébergement d'application en cluster

1. Objectifs du chapitre et prérequis 223
2. Déploiement d'une base de données MariaDB 223
   2.1 Origine du besoin 223
   2.2 Déploiement 224
      2.2.1 Choix de l'image Docker 224
      2.2.2 Version initiale du fichier de déploiement 224
      2.2.3 Gestion de la réentrance 225
   2.3 Volume persistant 226
      2.3.1 Demande de volume persistant 226
      2.3.2 État de la demande de volume persistant 226
      2.3.3 Ajout d'une persistance sur le conteneur de MariaDB . . 227
      2.3.4 Consultation de l'état du déploiement 228
   2.4 Configuration de la base de données 229
   2.5 Consultation de l'état du pod 230
      2.5.1 Liste des pods 230
      2.5.2 Connexion au conteneur 231
      2.5.3 Création de l'entrée de service 231
   2.6 Surveillance de la base de données 232
      2.6.1 Définition des commandes de surveillance 232
      2.6.2 Application de la modification 234
      2.6.3 Vérification du déploiement 234
   2.7 Mécanisme de déploiement 235
3. Mise en place d'un StatefulSet 237
   3.1 Augmentation du nombre de pods associés au déploiement . . 237
   3.2 Présentation du type StatefulSet 239
      3.2.1 Caractéristiques 239
      3.2.2 Limitations 239

3.3 Déclaration du premier objet StatefulSet 240
3.3.1 Purge de l'ancien déploiement 240
3.3.2 Modifications à réaliser 240
3.3.3 Création du StatefulSet 242
3.3.4 État des volumes persistants 242
3.3.5 Suppression des anciens objets PV/PVC 243
3.4 Scalabilité de l'objet StatefulSet 243
3.5 Pods et volumes persistants d'un objet StatefulSet 244
3.6 Réduction de la taille du StatefulSet 245
4. Base et compte de test 246
4.1 Variables d'environnement du conteneur 246
4.2 ConfigMap et secret 246
4.2.1 Pourquoi y faire appel ? 246
4.2.2 Structure d'un objet ConfigMap 247
4.2.3 Déclaration d'un objet Secret 247
4.2.4 Rattachement au conteneur 249

Chapitre 8
Mise en place d'une réplication entre pods

1. Objectifs du chapitre et prérequis 251
2. Synchronisation des pods MariaDB 252
2.1 Exposition de la problématique 252
2.2 Principe de fonctionnement de la synchronisation 252
2.2.1 Opérations à réaliser 252
2.2.2 Nombre de réplicas 253
2.3 Identifiants des serveurs 253
2.3.1 Connexion aux pods 253
2.3.2 Connexion à la base de données 253
2.3.3 Identifiants des serveurs 254
2.3.4 ID du maître 254
2.3.5 Création du compte de réplication sur le maître 255
2.3.6 Configuration de l'esclave 256

2.4 Activation de la synchronisation 256
2.4.1 Activer les journaux pour la réplication 256
2.4.2 Commande docker-entrypoint.sh 257
2.4.3 Consultation de l'état du maître 258
2.4.4 Configuration de l'esclave 259
2.5 Test de la réplication 260
2.5.1 Connexion au maître 260
2.5.2 Création d'une table 261
2.5.3 Connexion à l'esclave 261
3. Automatisation de la synchronisation 262
3.1 Scripts de démarrage et synchronisation 262
3.1.1 Script de démarrage 262
3.1.2 Configuration de la synchronisation 263
3.1.3 Scripts SQL additionnels 264
3.1.4 Script d'arrêt de la base 264
3.2 Scripts et objet ConfigMap 265
3.3 Création du ConfigMap 265
3.4 Montage du ConfigMap 268
3.4.1 Référencement du ConfigMap dans la liste des volumes 268
3.4.2 Point de montage du ConfigMap 269
3.5 Démarrage et arrêt du conteneur 269
3.5.1 Commande de démarrage 269
3.5.2 Commande d'arrêt de la base 270
3.6 Résumé des modifications 270
3.7 État du déploiement 272
3.7.1 État des pods 272
3.7.2 Journaux d'activité du pod esclave 272
3.7.3 Test de la synchronisation 272
3.7.4 Vérification du fonctionnement de la synchronisation . . 273

Chapitre 9
Gestion des briques internes de Kubernetes

1. Objectifs du chapitre et prérequis 275
2. Espace de noms kube-system 275
   2.1 Pods présents dans l'espace de noms kube-system 275
   2.2 CoreDNS 276
   2.3 etcd 277
   2.4 Le gestionnaire de réseau Kindnet 277
   2.5 Le gestionnaire d'extensions de Minikube 277
   2.6 Le serveur d'API 278
   2.7 Le proxy Kubernetes (kube-proxy) 278
   2.8 Le gestionnaire de tâches (scheduler) 278
   2.9 Le gestionnaire de contrôle (controller manager) 278
   2.10 Kubelet 278
3. Configuration des serveurs maîtres 279
   3.1 Principe de lancement des pods système 279
   3.2 Contenu du répertoire /etc/kubernetes/manifests 279
   3.3 Contenu des fichiers 280
   3.4 Désactivation d'un pod système 281
   3.5 Réactivation du pod système 282
4. Monitoring des conteneurs du cluster avec Glances 283
   4.1 Origine du besoin 283
   4.2 Consultation des DaemonSets 283
   4.3 Présentation de Glances 284
   4.4 Définition du DaemonSet 284
      4.4.1 Structure de la déclaration 284
      4.4.2 Champ volumes 285
      4.4.3 Champ containers 285
   4.5 Création du DaemonSet 286
      4.5.1 Déclaration complète 286
      4.5.2 Création du DaemonSet 287
      4.5.3 Consultation des pods 287

4.6 Annotations de tolérance 288
4.6.1 Présentation du mécanisme 288
4.6.2 Récupération des annotations taints 288
4.6.3 Tolérances de lancement 290
4.6.4 Modification du DaemonSet 290
4.7 Arrêt d'un DaemonSet 292
4.8 Connexion à Glances 294

Chapitre 10
Helm - Gestionnaire de package

1. Objectifs du chapitre et prérequis 295
2. Présentation de Helm 295
2.1 Pourquoi faire appel à Helm ? 295
2.2 Principe de fonctionnement 296
3. Déploiement de Helm 296
3.1 Installation du client Helm 296
3.1.1 Installation à l'aide d'Arkade 296
3.1.2 Installation manuelle 297
3.2 Consultation de la version de Helm 298
3.3 Configuration du client Helm 298
4. Déploiement d'une application avec Helm 299
4.1 Déterminer le package à déployer 299
4.1.1 Recherche d'un chart Helm 299
4.1.2 Gestion des sources de charts Helm 299
4.1.3 Recherche et gestion du cache de Helm 301
4.2 Installation du package WordPress 302
4.2.1 Un peu de vocabulaire 302
4.2.2 Lancement de l'installation 302
4.2.3 Installation sans accès direct 304

4.3 Corrections de l'installation 304
4.3.1 Quelques remarques 304
4.3.2 Spécification du nom et espace de noms 304
4.3.3 Lancement de l'installation 305
4.3.4 Mise à jour et réentrance 305
4.3.5 Généralisation des options d'installation 311
4.4 Éléments déployés avec Helm 311
4.5 Suppression d'un déploiement 312
4.6 Annulation de la suppression 313
4.7 Purge d'un chart Helm 314
5. Cycle de vie d'une application déployée avec Helm 315
5.1 Ouverture du port vers WordPress 315
5.2 Connexion à WordPress 316
5.3 Configuration d'un chart Helm 317
5.3.1 Consultation des options d'un chart 317
5.3.2 Configuration de la publication (Minikube) 319
5.4 Historique de déploiement 321
5.5 Visualisation des différences avant installation 322
5.5.1 Origine du besoin 322
5.5.2 Installation de l'extension diff pour Helm 322
5.5.3 Visualisation des modifications dans l'historique 324
5.5.4 Visualisation des modifications avant installation 325
5.6 Retour arrière 327
5.7 Helm Dashboard 328
5.8 Portail Artifact Hub 330

Chapitre 11
Contextes et outils tiers Kubernetes

1. Objectifs du chapitre et prérequis 333
2. Gestion des contextes avec kubectl 334
2.1 Origine du besoin 334
2.2 Lister les contextes 334

2.3 Variable d'environnement KUBECONFIG 335
2.3.1 Spécifier l'emplacement du fichier 335
2.3.2 Spécifier plusieurs fichiers 335
2.4 Changement de contexte 336
2.5 Créer un contexte 337
2.6 Supprimer un contexte 338
2.7 Outils de gestion de contexte 339
2.7.1 Présentation de kubectx et kubens 339
2.7.2 Installation de kubectx et kubens 339
2.7.3 Mise en place de l'autocomplétion 340
2.7.4 Test des commandes 341
2.8 Contexte dans le prompt utilisateur 341
2.8.1 Pourquoi afficher le contexte ? 341
2.8.2 Activation à l'aide de oh-my-zsh 342
2.8.3 Activation avec bash 342
2.8.4 Exemple d'affichage 342
2.9 Changement des couleurs du terminal 343
2.9.1 Principe de fonctionnement 343
2.9.2 Création de la fonction 344
2.9.3 Ajout de l'appel dans l'invite de commandes 345

3. Utilitaires Kubernetes 346
3.1 Krew : gestionnaire d'extensions 346
3.1.1 Présentation du mécanisme d'extensions 346
3.1.2 Installation de Krew 346
3.1.3 Test de l'extension 347
3.2 node-shell : lancer une session sur un nœud du cluster 348
3.2.1 Contexte 348
3.2.2 Installation de node-shell 349
3.2.3 Utilisation de node-shell 350
3.3 k9s : interface texte de suivi 350
3.3.1 Contexte 350
3.3.2 Installation de k9s 351
3.3.3 Lancement de k9s 351

3.3.4 Personnalisation de k9s 352
3.4 Kubespy : espionnage de l'activité 353
3.4.1 Présentation de Kubespy 353
3.4.2 Installation de Kubespy 354
3.4.3 Observation d'un déploiement avec Kubespy 354
3.5 Sniff : capture du trafic réseau d'un pod 355
3.5.1 Principe de fonctionnement 355
3.5.2 Installation de Sniff et Wireshark 355
3.5.3 Lancement d'une séance de capture 356

Chapitre 12
Analyse et sécurisation d'un cluster

1. Objectifs du chapitre et prérequis 359
2. Trivy : analyse de failles de sécurité 360
2.1 Trivy : outil d'analyse du cluster 360
2.1.1 Présentation de Trivy 360
2.1.2 Installation de Trivy 360
2.1.3 Lancement de l'analyse 361
2.1.4 Automatisation du lancement de l'analyse 363
2.1.5 Résultat de l'analyse 366
2.1.6 Lancement à intervalles réguliers 367
2.2 Mise en place de l'opérateur Trivy 370
2.2.1 Problèmes liés au rapport généré par Trivy 370
2.2.2 Installation de l'opérateur 370
2.2.3 Utilisation des nouveaux types de l'opérateur Trivy . . . 371
3. OPA (Open Policy Agent) Gatekeeper : le gardien du cluster 374
3.1 Présentation d'OPA Gatekeeper 374
3.2 Installation de Gatekeeper 374
3.3 Principe de fonctionnement 376
3.4 Création d'une règle OPA 376
3.5 Ressources à disposition de la communauté 380

Chapitre 13
Services managés Kubernetes

1. Objectifs du chapitre et prérequis 381
2. Service managé de Google : GKE 382
   2.1 Présentation du service Google 382
   2.2 Administration depuis la console Google 382
   2.3 Installation de la commande gcloud en local 384
      2.3.1 Installation sur Debian/Ubuntu 384
      2.3.2 Mise en place de l'autocomplétion 385
   2.4 Configuration de l'environnement 386
      2.4.1 Authentification auprès de Google Cloud 386
      2.4.2 Projet associé avec le contexte courant 388
      2.4.3 Activation de l'API 388
   2.5 Gestion du cluster GKE 389
      2.5.1 Consultation de la liste des clusters 389
      2.5.2 Versions et régions disponibles 390
   2.6 Création d'un cluster 390
      2.6.1 Options de création 390
      2.6.2 Lancement de la création du cluster 391
      2.6.3 Récupération du fichier d'accès au cluster 393
   2.7 Consultation du cluster 393
      2.7.1 Liste des nœuds 393
      2.7.2 Services démarrés 393
   2.8 Délégation des droits d'accès 396
      2.8.1 Configuration des accès 396
      2.8.2 Principe du mécanisme sous-jacent 397
   2.9 Suppression d'un cluster GKE 398
3. Service managé Microsoft Azure : AKS 399
   3.1 Présentation du service Azure 399
   3.2 Administration depuis la console Azure 399
      3.2.1 Présentation de la console 399
      3.2.2 Consultation du tableau de bord Kubernetes 401

3.3 Installation de la commande az en local 401
3.3.1 Installation sur Debian/Ubuntu 402
3.3.2 Mise en place de l'autocomplétion 402
3.4 Authentification auprès du service Azure. 403
3.5 Emplacement de déploiement 404
3.5.1 Liste des emplacements 404
3.5.2 Versions disponibles de Kubernetes 405
3.6 Création d'un cluster 406
3.6.1 Création d'un groupe de ressources. 406
3.6.2 Lancement de la création du cluster 407
3.6.3 Récupération du fichier de connexion. 409
3.6.4 Zone DNS par défaut 409
3.7 Consultation de la liste des clusters 410
3.8 Délégation des droits d'accès 410
3.9 Suppression d'un cluster AKS 411
4. Service managé d'Amazon : EKS 411
4.1 Présentation du service Amazon AWS 411
4.2 Introduction de la commande eksctl. 412
4.3 Configuration des accès Amazon 412
4.4 Installation des binaires 416
4.4.1 Installation d'eksctl 416
4.4.2 Installation de l'outil aws cli 416
4.4.3 Vérification de la communication avec AWS 417
4.5 Création du cluster EKS 418
4.5.1 Aide en ligne d'eksctl 418
4.5.2 Options intéressantes à la création d'un cluster. 419
4.5.3 Lancement de la création du cluster 420
4.6 Configuration des accès kubectl 424
4.7 Mécanisme de communication 424
4.8 Délégation des droits d'accès 425
4.8.1 Configuration des accès 425
4.8.2 Principe du mécanisme sous-jacent. 426
4.9 Suppression du cluster 427

5. Service Kubernetes OVHcloud 429
5.1 Présentation d'OVHcloud 429
5.2 Méthode de création du cluster et prérequis 429
5.3 Réseau des nœuds du cluster 430
5.4 Création du cluster Kubernetes 431
5.5 Suppression du cluster Kubernetes 432
6. Infrastructure as Code 433
6.1 Origine du besoin 433
6.2 Installation d'OpenTofu 433
6.3 Génération des identifiants d'accès OVH 434
6.4 Déclaration des objets dans OpenTofu/Terraform 437
6.5 Téléchargement des dépendances 441
6.6 Création de la plateforme 442
6.7 Test et utilisation du cluster 444
6.8 Destruction du cluster 444
7. Accès en lecture-écriture multiple 446
7.1 Origine du besoin 446
7.2 Serveur NFS déployé dans Kubernetes 446
7.2.1 Limitations 446
7.2.2 Déploiement d'un serveur NFS 447
7.2.3 Vérification du déploiement 448
7.2.4 Test de création de volume persistant (PVC) 448
7.3 Service managé Amazon : EFS 449
7.3.1 Consultation des instances EFS 449
7.3.2 Création d'une instance EFS 449
7.4 Service managé Google : Filestore 450
7.4.1 Consultation des instances Filestore 450
7.4.2 Création d'une instance Filestore 451
7.5 Classe de stockage NFS 452
7.5.1 Installation du chart 452
7.5.2 Vérification de l'installation du chart 452
7.5.3 Test de création d'une demande de volume persistant (PVC) 453

7.6 Classe de stockage Azure 453

Chapitre 14
Installation de Kubernetes en interne

1. Objectifs du chapitre et prérequis 455
2. Installation à l'aide de Kubespray 456
2.1 Origine du besoin 456
2.2 Pourquoi Kubespray ?. 456
2.3 Principe de Kubespray 456
2.4 Prérequis des machines à administrer avec Ansible 457
2.4.1 Échange de clés SSH 457
2.4.2 Escalade de droits sudo. 457
2.5 Structure du cluster 459
2.5.1 Architecture du cluster Kubespray 459
2.5.2 Groupes de machines 459
2.6 Préparation des éléments d'installation 460
2.6.1 Clonage du dépôt Git 460
2.6.2 Installation des prérequis 460
2.7 Préparation de l'inventaire Ansible 461
2.7.1 Qu'est-ce qu'un inventaire ? 461
2.7.2 Fichier d'inventaire d'exemple. 461
2.7.3 Test de la communication Ansible/SSH 463
2.8 Installation du cluster 464
2.8.1 Configuration du cluster 464
2.8.2 Lancement de l'installation 465
2.8.3 Mise à jour du cluster 465
2.9 Accès au cluster. 466
2.9.1 Configuration de l'accès au cluster 466
2.9.2 Utilisation d'un répartiteur de charge. 466
2.9.3 Vérification de la communication avec le cluster. 467

3. Environnement embarqué avec k3s. 468
3.1 Présentation et but du projet. 468
3.2 Installation de k3s 469
3.3 Communication avec le cluster 470
3.4 Ajout de nœuds au cluster 471

Chapitre 15
Exposition des applications sur Internet

1. Objectifs du chapitre et prérequis 473
2. Gestion des entrées DNS 474
2.1 Principe de fonctionnement. 474
2.2 Prérequis 475
2.3 Retour sur le fonctionnement du domaine DNS nip.io. 475
2.4 Activation du service DNS. 476
2.4.1 Console Cloud DNS Google 476
2.4.2 Service DNS Zones d'Azure. 479
2.4.3 Service Route 53 d'AWS. 480
2.5 Configuration DNS 481
2.5.1 Opération à réaliser 481
2.5.2 Console DNS OVH 481
2.5.3 Vérification de la délégation 483
2.6 Gestionnaire de DNS 484
2.6.1 Présentation de la brique external-dns 484
2.6.2 Création du compte d'administration DNS de Google Cloud 484
2.6.3 Création du compte d'administration DNS service Azure. 486
2.6.4 Création du compte d'administration DNS Amazon (Route 53) 488
2.6.5 Création d'un secret (Google et Azure). 490
2.6.6 Déploiement d'external-dns 491
2.6.7 Vérification du fonctionnement d'external-dns. 493

3. Exposition de services et répartition de charge 494
3.1 Présentation du mécanisme 494
3.2 Retour sur la notion de service 494
3.2.1 Rôle d'un service 494
3.2.2 Structure d'un service 495
3.2.3 Type ClusterIP 495
3.2.4 Type NodePort et LoadBalancer 496
3.2.5 Type ExternalName 496
3.3 Service associé au proxy inverse 496
4. Le contrôleur Ingress 497
4.1 Principe de fonctionnement 497
4.2 Le rôle du contrôleur Ingress 498
4.3 Structure d'une règle Ingress 498
4.4 Droits nécessaires pour un contrôleur 500
5. Le contrôleur Ingress Google 500
5.1 Prérequis 500
5.2 Présentation du contrôleur GLBC 501
5.3 Déploiement de Mailpit 501
5.3.1 Préparation de la règle Ingress 501
5.3.2 Déploiement de l'application Mailpit 502
5.3.3 Consultation de l'état de l'objet Ingress 502
5.3.4 Consultation du journal d'activité d'external-dns 502
5.3.5 Consultation de Mailpit 504
6. Le contrôleur Ingress Nginx 507
6.1 Pourquoi changer de contrôleur Ingress ? 507
6.2 Présentation du logiciel Nginx 507
6.3 Installation d'Ingress Nginx sur GKE (Google) 508
6.3.1 Détermination du chart Helm à installer 508
6.3.2 Espace de noms et configuration du chart 508
6.4 Utilisation du contrôleur 509
6.4.1 Utilisation du champ spec.ingressClassName 509
6.4.2 Vérification du déploiement 510

6.5 Annotations Ingress Nginx 511
6.6 Ajustement de la configuration du contrôleur Nginx 512
6.6.1 Récupération de l'origine du trafic 512
6.6.2 Haute disponibilité et règles d'affinité/anti-affinité . . . 514
6.6.3 Gestion des contrôleurs Ingress 517
6.6.4 Réservation et limitation des ressources disponibles . . . 519
7. Le contrôleur Ingress Traefik 521
7.1 Présentation de Traefik 521
7.2 Installation du chart Helm 521
7.3 Utilisation du nouveau contrôleur Ingress 524
7.3.1 Sélectionner le contrôleur Ingress Traefik 524
7.3.2 Création de la règle Ingress faisant appel à Traefik 524
7.3.3 État des règles Ingress. 525
7.4 Tableau de bord de Traefik 526
7.5 Annotations Ingress Traefik 528
7.6 Distribution de la charge et haute disponibilité 529
7.7 Réservation et limitation des ressources disponibles 530
7.8 Pour aller plus loin 530

Chapitre 16
Sécurisation : accès aux applications

1. Objectifs du chapitre et prérequis 533
2. Mise en place de Let's Encrypt. 534
2.1 Présentation de Let's Encrypt 534
2.2 Installation du chart Helm cert-manager 534
2.2.1 Présentation du chart Helm. 534
2.2.2 Prérequis avant installation 535
2.3 L'émetteur de certificats (issuer) 536
2.3.1 Principe du protocole ACME 536
2.3.2 Structure de la déclaration d'un émetteur 536
2.3.3 Exemple de déclaration Issuer Google. 538
2.3.4 Exemple de déclaration Issuer Azure. 539

2.3.5 Exemple de déclaration Issuer Amazon 540
2.3.6 Limitations et certificats avec joker 540
2.4 Exemples de déclarations 541
2.4.1 Serveur Let's Encrypt de test 541
2.4.2 Serveur Let's Encrypt de production 542
2.5 Déclaration des certificats 543
2.5.1 État des émetteurs de certificats 543
2.5.2 Structure d'un certificat 543
2.5.3 Certificat de test 544
2.5.4 État du certificat 545
2.5.5 Journal d'activité de cert-manager 545
2.5.6 Consultation du secret 547
2.5.7 Certificat de production 549
2.5.8 Marche à suivre en cas de problèmes 550
2.6 Rattachement du certificat à la règle Ingress 550
2.7 Automatisation de la gestion des certificats 552
2.7.1 Certificat par défaut du contrôleur Ingress Nginx 552
2.7.2 Mécanisme d'annotations 553
2.7.3 Émetteur de certificats par défaut 554

3. Protection de l'accès aux applications 555
3.1 Origine du besoin 555
3.2 Mot de passe simple (HTTP basic) 555
3.2.1 Principe de fonctionnement 555
3.2.2 Création du secret à l'aide de htpasswd 556
3.2.3 Import du secret 556
3.2.4 Configuration de l'authentification 557

4. Authentification basée sur OAuth2 561
4.1 À propos du protocole OAuth2 561
4.2 Principe de la solution 561
4.3 Création d'un identifiant GitHub 562
4.4 Déploiement du proxy 564
4.4.1 À propos du proxy 564
4.4.2 Configuration du chart Helm 565

4.4.3 Déploiement du chart Helm 566
4.4.4 État du déploiement 566
4.5 Déclaration des règles Ingress 567
4.5.1 Description des règles Ingress 567
4.5.2 Annotations Ingress de Mailpit 567
4.5.3 Description Ingress du proxy OAuth 567
4.5.4 Déclaration des règles Ingress 567
4.6 Tests de connexion 569
4.7 Restriction des accès 570
4.7.1 Mécanisme d'autorisation 570
4.7.2 Restriction par domaine e-mail 571
4.7.3 Restriction par organisation GitHub 571

Chapitre 17
Polices réseau

1. Objectifs du chapitre et prérequis 573
2. Les polices réseau (network policies) 573
2.1 Présentation du mécanisme 573
2.2 Kubernetes Network Plugins 574
2.3 Polices réseau sur services managés et installation maison . . . 575
2.4 Installation de Calico sur Minikube 575
2.4.1 Activation de CNI sur Minikube 575
2.4.2 Installation de Calico 575
2.5 Connexions entrantes 576
2.5.1 Test de la connexion en interne 576
2.5.2 Bloquer les accès internes 576
2.5.3 Test de la règle 578
2.6 Connexions sortantes 579
2.6.1 Test des connexions externes 579
2.6.2 Restriction sur les règles sortantes 579
2.6.3 Test de la règle 580

3. Protection de l'application WordPress. 580
3.1 Contexte 580
3.2 Déploiement de WordPress 580
3.3 Restriction des accès. 582
3.3.1 Référencement de l'ensemble des flux 582
3.3.2 Restriction de tous les accès. 582
3.3.3 Autorisation de l'accès du contrôleur Ingress sur WordPress 583
3.3.4 Autorisation de l'accès entre WordPress et MariaDB . . . 584
3.3.5 Test des règles réseau 585
3.4 Ressources externes 587

Chapitre 18
Montée en charge automatique

1. Objectifs du chapitre et prérequis 589
2. Le serveur de métriques 590
2.1 Présentation de la brique metrics-server 590
2.2 Activation du serveur de métriques. 590
2.2.1 Vérification de la présence du serveur de métriques . . . 590
2.2.2 Activation sous Minikube 591
2.2.3 Activation sur Amazon EKS 591
2.3 Consultation de la consommation des pods et des nœuds . . . 592
2.3.1 État des nœuds d'un cluster. 592
2.3.2 État des pods. 592
3. Activation de la montée en charge automatique 593
3.1 Test avec l'application Mailpit 593
3.2 Lancement d'un bench. 594
3.2.1 Présentation d'Apache Bench 594
3.2.2 Installation d'Apache Bench 595
3.2.3 Lancement du test initial. 595
3.3 Gestion de la montée en charge. 596

3.4 Lancement du test 597
3.4.1 Test à froid : montée en charge des pods 597
3.4.2 Test à chaud 598
3.4.3 Diminution du nombre de pods 599
4. Scalabilité des nœuds d'un cluster. 599
4.1 Contexte 599
4.2 Présentation de l'autoscaler 600
4.3 Activation de l'autoscaler avec Google Cloud 600
4.4 Activation de l'autoscaler avec Azure AKS 601
4.5 Activation de l'autoscaler avec Amazon EKS 601
4.5.1 Présentation du mécanisme de l'ASG 601
4.5.2 Affichage des tags d'un groupe ASG 602
4.5.3 Activation de l'autoscaler. 603
4.5.4 Vérification de l'activation du mécanisme de l'autoscaler 604
4.6 Test de montée en charge 604
4.6.1 Déploiement de WordPress 604
4.6.2 Pods en attente de ressources. 605
4.6.3 État des nœuds 605
4.6.4 Démarrage du pod 606
4.6.5 Nettoyage des déploiements 606
5. Définition de groupes de machines 607
5.1 Contexte 607
5.2 Les différents types de machines 608
5.2.1 Les machines préemptives (Spot) 608
5.2.2 Réservation d'instances 609
5.2.3 Familles de machines 609
5.2.4 Quelques exemples de familles de machines 610
5.2.5 Utilisation d'accélérateurs graphiques (GPU). 612
5.2.6 Quelques exemples de coûts 613
5.3 Création de groupes de machines 615
5.3.1 Contexte 615
5.3.2 Création d'un groupe avec Google 616

5.3.3 Création d'un groupe avec Azure 617
5.3.4 Création d'un groupe avec AWS EKS 619
5.3.5 Création d'un groupe avec OVH 621
5.4 Mécanisme de tolérance et sélection de nœuds 623

Chapitre 19
Surveillance à l'aide de Prometheus

1. Objectifs du chapitre et prérequis 627
2. Mise en place de Prometheus 628
2.1 À propos de Prometheus 628
2.2 Fonctionnement de Prometheus 629
2.2.1 Architecture de Prometheus 629
2.2.2 Le moteur Prometheus 629
2.2.3 Les exporteurs Prometheus 630
2.3 Installation de Prometheus 631
2.3.1 Choix du chart Prometheus 631
2.3.2 Qu'est-ce qu'un opérateur ? 631
2.3.3 Déploiement de l'opérateur Prometheus 632
2.3.4 Pods démarrés 634
2.3.5 Objets déploiements 634
2.3.6 Nouvelles ressources Prometheus 635
2.3.7 DaemonSet : node exporter 636
2.4 Priorisation des briques de surveillance 637
2.4.1 Problème de la surveillance 637
2.4.2 Déclaration des classes de priorité 637
2.4.3 Modification du déploiement de Prometheus 638
3. Utilisation de Prometheus 638
3.1 Fonctionnement des métriques 638
3.1.1 Consultation des métriques de Prometheus 638
3.1.2 Présentation de l'interface de Prometheus 639
3.1.3 Métriques de Kubernetes 641
3.1.4 Déclaration des points de collecte dans Kubernetes 642

3.1.5 Consultation des points de collecte dans Prometheus . . 643
3.2 Définition des alertes 644
3.2.1 Consultation de la liste des alertes 644
3.2.2 Structure d'une règle d'alerte 645
3.2.3 Définition d'alertes 646
3.3 Gestionnaire d'alertes 647
3.3.1 Rôle du gestionnaire d'alertes 647
3.3.2 Consultation du gestionnaire d'alertes 648
3.3.3 Configuration des alertes 648
3.3.4 Désactivation des alertes scheduler et manager (clusters managés) 649
3.3.5 Configuration de l'envoi des notifications 651
3.3.6 Adresse de l'API de Slack 652
4. Tableaux de bord Grafana 655
4.1 Présentation de Grafana 655
4.2 Configuration de Grafana 655
4.2.1 Branchement au moteur Prometheus 655
4.2.2 Définition des tableaux de bord 656
4.3 Interface Grafana 656
4.4 Sécurisation de l'accès à Grafana 658
5. Suppression du chart de Prometheus 659

Chapitre 20
Centralisation des journaux d'activité

1. Objectifs du chapitre et prérequis 661
2. Principe de la centralisation des journaux 662
2.1 Architecture 662
2.2 Caractéristiques de l'agent déployé 663
2.3 Mécanisme de centralisation des journaux 664

3. Centralisation dans le cloud 664
3.1 Centralisation à l'aide d'un service managé 664
3.2 Google Stackdriver 665
3.2.1 Présentation de Stackdriver 665
3.2.2 Pod Fluent Bit (cluster GKE) 665
3.2.3 Consultation des journaux 665
3.3 Azure Monitor 666
3.3.1 Présentation d'Azure Monitor 666
3.3.2 Consultation des journaux 667
3.4 Amazon Cloudwatch 668
3.4.1 Présentation de Cloudwatch 668
3.4.2 Activation de Cloudwatch sur le centre de contrôle . . . 668
3.4.3 Configuration de Cloudwatch 669
3.4.4 Création de la police de communication avec Cloudwatch 670
3.4.5 Création d'un compte et rattachement à la police 671
3.4.6 Création d'une clé d'accès 672
3.4.7 Envoi des journaux dans Cloudwatch 672
3.4.8 État des pods déployés 674
3.4.9 Consultation des journaux dans Cloudwatch 675
4. Centralisation des journaux avec Loki 676
4.1 Présentation de Loki 676
4.1.1 Origine de Loki 676
4.1.2 Loki vs Opensearch/Elasticsearch 676
4.1.3 Conseil d'utilisation 676
4.2 Installation de Loki 677
4.3 Configuration de la source de données Grafana 678
4.4 Consultation des journaux dans Grafana 680
4.4.1 Vérification des sources de données 680
4.4.2 Consultation des journaux 680
5. Centralisation des journaux avec Opensearch/Elasticsearch 682
5.1 Avertissements et limitations 682
5.2 À propos d'Opensearch et d'Elasticsearch 682

5.3 Déploiement des briques Opensearch/Elasticsearch 683
5.3.1 Installation d'Opensearch 683
5.3.2 Installation de l'agent Fluent-bit 684
5.3.3 Installation de Kibanad/Opensearch Dashboards 687
5.4 État des différentes briques 687
5.4.1 État du moteur Elasticsearch 687
5.4.2 Agent Fluent-bit 688
6. Gestion d'Opensearch/Elasticsearch 688
6.1 Utilisation de Kibana/Opensearch Dashboards 688
6.1.1 Accéder à l'interface de Kibana ou Opensearch Dashboards 688
6.1.2 Création de l'index 690
6.2 Branchement sur Grafana 692
6.2.1 Source de données Opensearch/Elasticsearch 692
6.2.2 Création d'un objet ConfigMap 692

Chapitre 21
Maillage de services

1. Objectifs du chapitre et prérequis 695
2. Présentation d'Istio 696
2.1 Micro-services et mise en réseau de services 696
2.2 Présentation d'Istio 697
2.3 Principe de fonctionnement 697
3. Installation d'Istio 699
3.1 Configuration d'external-dns 699
3.2 Dépôt des charts Istio 700
3.3 Installation d'istio 701
3.4 Configuration d'Istio 701
3.4.1 Activation des tableaux de bord 701
3.4.2 Configuration d'Istio 702

3.5 Configuration du composant Gateway Istio 703
3.5.1 Présentation du mécanisme 703
3.5.2 Installation du composant Gateway 703
3.5.3 Génération du certificat 705
4. Utilisation d'Istio 706
4.1 Injection de pods dans le maillage de services 706
4.1.1 Installation d'istioctl 706
4.1.2 Injection du sidecar à l'aide d'istioctl 707
4.1.3 Injection du sidecar par annotation 708
4.1.4 Désactivation du sidecar par annotation 708
4.2 Déploiement d'une application de test 709
4.2.1 Principe de l'exposition d'application avec Istio 709
4.2.2 Préparation du fichier de déploiement Mailpit 710
4.2.3 Déclaration du service Mailpit 713
4.2.4 Création de l'objet Gateway 713
4.2.5 Création du service virtuel (VirtualService) 715
4.3 Sécurisation des flux 716
4.3.1 Activation de Mutual TLS (mTLS) 716
4.3.2 Consultation des polices du service Mailpit 717
4.3.3 Forcer l'utilisation de mTLS 719
5. API Gateway 722
5.1 Origine du besoin 722
5.2 Principe de fonctionnement 723
5.3 Installation de l'API Gateway 723
5.4 Définition de l'objet Gateway 724
5.5 Définition de l'objet HTTPRoute 728
5.6 Configuration du module external-dns 730
5.7 Un mot sur les limitations actuelles 731
6. Tableaux de bord 731
6.1 Présentation des différentes briques 731
6.2 Interface Kiali 732
6.3 Interface Grafana 733

6.4 Interface Jaeger 736

Chapitre 22
Compilation et stockage d'images Docker

1. Objectifs du chapitre et prérequis 739
2. Création d'une image Docker 740
   2.1 Application d'exemple Flask HealthCheck 740
      2.1.1 Présentation de l'application 740
      2.1.2 Présentation des dépendances 740
      2.1.3 Description des dépendances 740
      2.1.4 Installation des dépendances 741
      2.1.5 Initialisation de l'application 742
      2.1.6 Fonction racine 742
      2.1.7 Lancement du programme 742
   2.2 Environnement de compilation 745
   2.3 Un mot sur les alternatives à Docker 745
   2.4 Le fichier Dockerfile 746
      2.4.1 Présentation du format 746
      2.4.2 Création de l'image 747
      2.4.3 Compilation et tag de l'image 748
      2.4.4 Authentification sur le registre d'images Docker 749
      2.4.5 Pousser l'image sur le registre 749
3. Image Docker multi-étape (multi-stage) 751
   3.1 Origine du besoin 751
   3.2 Exemple de compilation avec Maven 751
4. Analyse d'images 752
   4.1 Historique des commandes 752
   4.2 Analyse de l'image : Dive 753
      4.2.1 Présentation de Dive 753
      4.2.2 Installation de Dive 753
      4.2.3 Consultation du contenu d'une image 753

5. Analyse de failles de sécurité avec Trivy 754
5.1 Origine du besoin 754
5.2 Installation de Trivy 755
5.3 Lancement de l'analyse 755
5.4 Configuration de l'analyse 756
6. Utilisation de registres Docker privés 758
6.1 Origine du besoin 758
6.2 Déploiement d'image d'un registre privé 758
6.2.1 Exposition de la problématique 758
6.2.2 Création du secret 758
6.2.3 Utilisation du secret 759
6.2.4 Utilisation d'un compte de service 760
6.2.5 Recopie d'un secret entre espaces de noms 762
6.3 Erreurs de récupération des images 763
6.3.1 Détection des erreurs 763
6.3.2 Erreur sur le nom de l'image 763
6.3.3 Secret absent ou non spécifié 764
6.3.4 Identifiants invalides 764
6.4 Services de registres managés 764
6.4.1 Solutions managées 764
6.4.2 Service Google Container Registry 765
6.4.3 GitLab.com 767

Chapitre 23
Usine logicielle

1. Objectifs du chapitre et prérequis 769
2. Compilation à l'aide de GitLab 770
2.1 Application à compiler 770
2.2 Mécanisme de pipeline GitLab 770
2.3 Adresse et contenu du dépôt 770
2.4 Structure du fichier .gitlab-ci.yml 771
2.5 Exemple de fichier .gitlab-ci.yml de compilation d'image 772

2.6 Construction d'image à l'aide de Kaniko 774
2.7 Pour la suite 776
3. Déploiement continu avec Jenkins 776
3.1 À propos de Jenkins 776
3.2 Installation de Jenkins 777
3.2.1 Configuration du chart 777
3.2.2 Vérification de l'installation 778
3.2.3 Connexion à l'interface de Jenkins 779
3.2.4 Installation d'extensions 779
4. Pipeline de déploiement continu avec Jenkins 780
4.1 Prérequis 780
4.2 Présentation du mécanisme de déploiement continu 780
4.3 Stockage des identifiants Docker 781
4.4 Création de l'environnement develop 781
4.5 Création du pipeline 782
4.5.1 Création du pod de compilation 782
4.5.2 Squelette du pipeline de déploiement 784
4.5.3 Récupération du code source 785
4.5.4 Vérifications et tests 785
4.5.5 Compilation de l'image Docker 786
4.5.6 Connexion au registre et publication 786
4.5.7 Mise à jour du déploiement de test 787
4.5.8 Programme complet 788
4.6 Lancement de la compilation 790
4.6.1 Création du job 790
4.6.2 Lancement du build 791
4.7 Compte de service 793
4.7.1 Opérations à réaliser 793
4.7.2 Création d'un compte de service 793
4.7.3 Création du rôle de mise à jour 793
4.7.4 Affectation du rôle au compte de service 794
4.7.5 Affectation du compte de service 795
4.7.6 Relance de la compilation 795

4.8 Utilisation de Kaniko avec Jenkins 796
4.9 Mécanisme de Webhook 798
4.9.1 Présentation du mécanisme 798
4.9.2 Déclenchement du Webhook. 798
4.9.3 Création du Webhook 799
5. Un mot sur Jenkins X. 800

Chapitre 24
Packager son application avec Helm

1. Objectifs du chapitre et prérequis 801
2. Helm 802
2.1 Origine du besoin 802
2.2 Création d'un chart 802
2.3 Contenu d'un package 803
2.3.1 Structure d'un package. 803
2.3.2 Variables .Values, .Chart et .Release 803
2.4 Adaptation à l'application Mailpit 804
2.4.1 Résumé des travaux à réaliser 804
2.4.2 Utilisation de l'image de Mailpit 804
2.4.3 Correction sur les ports de service. 804
2.4.4 Ajout d'un ConfigMap. 806
2.4.5 Ajout du volume persistant 806
2.4.6 Montage du volume dans le conteneur. 807
2.4.7 Test de déploiement 808
2.5 Dépendances 809
2.5.1 Déclaration des dépendances 809
2.5.2 Récupération des dépendances 810
2.5.3 Déploiement du chart avec les dépendances. 811
3. Template Go 812
3.1 Principe de fonctionnement. 812
3.2 Substitution du contenu d'une variable 812

3.3 Blocs conditionnels 813
3.4 Gestion des conditions 813
3.4.1 Les valeurs de vrai ou faux 813
3.4.2 Opérateurs de comparaison 814
3.4.3 Conditions multiples et négation 815
3.5 Itération 815
3.5.1 Affichage du contenu d'un tableau 815
3.5.2 Suppression des sauts de lignes surnuméraires 816
3.5.3 Itération sur un groupe fixe d'éléments 817
3.5.4 Itération sur un tableau de hachage 817
3.5.5 Accéder à une variable globale depuis une boucle 818
3.6 Filtres 818
3.7 Valeurs par défaut 819
3.8 Fonction template 819
3.8.1 Exemple de définition de fonction 819
3.8.2 Passage de paramètres et contexte global 820
3.9 Redéploiement sur changement de configuration 821
3.9.1 Exposition de la problématique 821
3.9.2 Principe de la solution 821
3.9.3 Exemple d'implémentation 821
3.10 Génération de secret aléatoire « stable » 823
3.10.1Exposition de la problématique 823
3.10.2Principe de la solution et mise en œuvre 823

Chapitre 25

Restriction et délégation d'accès

1. Objectifs du chapitre et prérequis 825
2. Mise en place de quotas 826
2.1 Origine du besoin 826
2.2 Quotas par défaut sur un espace de noms 826
2.2.1 Création d'un espace de noms 826
2.2.2 Structure d'un objet LimitRange 827

2.2.3 Exemple de définition de limitations 828
2.2.4 Vérification de l'application des limitations 829
2.2.5 Test du mécanisme 830
2.2.6 Analyse du problème 830
2.3 Quotas globaux sur un espace de noms 831
2.3.1 Présentation des quotas de ressources (ResourceQuota) . 831
2.3.2 Structure d'un quota de ressources 832
2.3.3 Exemples de restriction de consommation CPU et mémoire 832
2.3.4 Champs pour positionner des limitations (champ hard) 833
2.3.5 Test du mécanisme 834
2.3.6 Nettoyage en fin d'exercice 835
3. Authentification et autorisation 836
3.1 Origine du besoin 836
3.2 Prérequis 837
3.3 Activation des accès anonymes 838
3.3.1 Activation des accès anonymes sur Minikube 838
3.3.2 Création du fichier d'accès 839
3.3.3 Affectation des droits en lecture 840
3.3.4 Suppression des droits d'accès anonymes 841
3.4 Principe de l'authentification par certificat 841
3.5 Problème de la révocation des certificats 842
3.6 Génération du certificat 842
3.6.1 Création de la clé et du certificat client 842
3.6.2 Emplacement de la PKI 843
3.6.3 Signature du certificat 844
3.7 Authentification par certificat 845
3.7.1 Récupération des informations de connexion au cluster 845
3.7.2 Utilisation du certificat pour l'authentification 846
3.7.3 Test de la connexion 848

3.8 Quelques exemples d'erreurs de manipulation 848
3.8.1 Problème d'autorité de certification 849
3.8.2 Problème de couple clé/certificat 849
3.8.3 Pas d'identifiant renseigné 849
3.8.4 Utilisation de la demande de certificat à la place du certificat 849
3.9 Attributions de droits administrateur sur le cluster 850
3.9.1 Contexte et opérations à réaliser 850
3.9.2 Affectation des droits administrateur à un utilisateur . 850
3.9.3 Test du nouvel administrateur 852
3.9.4 Création de nouveaux utilisateurs 852
3.9.5 Affectation des droits administrateur à un groupe 853
3.9.6 Administrateur d'un espace de noms 853

4. Mécanismes d'authentification externes 855
4.1 Présentation du mécanisme 855
4.2 Communication entre le fournisseur OAuth2 et le cluster . . . 856
4.3 Création des identifiants 857
4.4 Modification des options de démarrage (Minikube) 858
4.5 Configuration des accès clients 860
4.5.1 Présentation de l'outil kubelogin/oidc-login 860
4.5.2 Installation de kubelogin/oidc-login 860
4.5.3 Génération des identifiants d'accès 860
4.5.4 Renseignement du cluster et contexte 863
4.5.5 Test de connexion 864
4.6 Attribution des droits d'accès 865

5. Mise en place de comptes de service 865
5.1 Contexte 865
5.2 Création du compte de service 866
5.3 Attribution des droits d'administrateur au compte de service . . 868
5.4 Création du fichier Kubeconfig 869
5.4.1 Recopie des informations de connexion 869
5.4.2 Jeton de connexion du compte de service 870
5.4.3 Affectation de l'utilisateur 871

5.4.4 Test de la connexion 872
5.4.5 Révocation des accès 872

Chapitre 26
Les opérateurs Kubernetes

1. Objectifs du chapitre et prérequis 875
2. Utilisation des opérateurs 876
2.1 Présentation du principe 876
2.2 L'opérateur de Prometheus 876
2.2.1 Retour sur le chart prometheus-operator 876
2.2.2 Structure d'un objet Prometheus 877
2.3 Ressources externes sur les opérateurs existants 878
2.4 Présentation des opérateurs MySQL et MariaDB 879
2.4.1 Choix de l'opérateur 879
2.4.2 Déploiement de l'opérateur 879
2.5 Création d'une instance MariaDB 880
2.5.1 Création du cluster 881
2.5.2 Problèmes de droits sur le volume persistant 882
2.6 Objets créés au déploiement du cluster 882
2.6.1 Volumes persistants 883
2.6.2 Services MariaDB 883
2.7 Test de la réplication 883
2.7.1 Initialisation de l'environnement de test 883
2.7.2 Connexion aux instances maître et esclave 885
2.7.3 Création d'une table 886
2.7.4 Alimentation de la table 886
2.7.5 Changement du nombre de réplicas 887
2.8 Pour conclure 888

Annexes

1. bash vs zsh 889
   1.1 Les shells Unix 889
   1.2 zsh et oh-my-zsh 890
   1.3 Tester zsh et oh-my-zsh. 890
   1.4 Configurer zsh comme shell pour l'utilisateur courant 891
2. Déploiement du tableau de bord Kubernetes 892
   2.1 Installation du tableau de bord (application dashboard) 892
   2.2 Création du compte d'accès. 893
   2.3 Récupération du jeton de connexion 894
   2.4 Décodage du contenu du jeton 894
   2.5 Connexion au tableau de bord. 896

Index 899