Editions ENI
Référentiel de compétences

Intitulé de la certification
Docker *Référentiel de mai 2023*
Description de la situation professionnelle à partir de laquelle le dispositif de formation visant la certification est initié :
Docker est une plateforme logicielle permettant de créer, de déployer et de gérer des d'applications virtualisées (conçues en micro services) sur un système d'exploitation. Les services ou fonctions de l'application et ses différentes bibliothèques, fichiers de configuration, dépendances et autres composants sont regroupés au sein du conteneur. Chaque conteneur exécuté partage les services du système d'exploitation. L'utilisation de Docker permet aux développeurs d'être sûrs que leur application fonctionnera indépendamment du système d'exploitation et de l'environnement auquel il est soumis. De l'autre, Docker permet aux administrateurs et techniciens système et réseau d'installer et de démarrer des conteneurs et donc de déployer toute une application et ses dépendances avec une seule commande. Docker permet aussi la simplification de l'installation de mises à jour par une configuration facile à implémenter. La maîtrise de la plateforme Docker présente un fort enjeu d'avenir pour les entreprises qui déploient et développent des applications. Cette plateforme permettant un gain d'efficacité et de productivité pour les développeurs, les administrateurs et les techniciens. Face à la popularité grandissante de Docker, les recruteurs sont donc de plus en plus à la recherche de professionnels maîtrisant cette plateforme de conteneurisation. La certification IT Docker démontre la compétence du candidat à utiliser Docker pour déployer des applications conçues en microservices.. En fonction du score obtenu à l'épreuve, les candidats valident la certification : o **De 0 à 699/1000** : le candidat n'est pas certifié o **De 700 à 1000/1000** : le candidat est certifié En dessous de 700 points, la certification n'est pas validée.

Référentiel de compétences	Référentiel d'évaluation	
COMPETENCES	**MODALITE D'EVALUATION**	**CRITERES D'EVALUATION**
C1 - Configurer Docker sur différents systèmes d'exploitation en utilisant Docker Desktop ou Docker Engine pour l'exploiter en environnement de développement, de test et de production.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr1_1** Les prérequis d'installation sont vérifiés pour l'installation de Docker (version du système d'exploitation, correctifs pour faire fonctionner Docker en mode WSL 2, CPU, RAM, espace disque disponible) **Cr1-2** L'installation de Docker est réalisée à l'aide de Docker Desktop et Docker Engine **Cr1_3** Les permissions d'utilisation sont correctement affectées (groupes docker-users et docker) **Cr1_4** La configuration pour un démarrage de Docker à l'amorçage du système est réalisée (Docker Desktop Service ou daemon Docker)
C2 - Utiliser les images, les registres et les conteneurs avec la ligne de commandes Docker pour déployer et gérer des services applicatifs.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr2_1** Une image est téléchargée d'un registre afin de créer un conteneur (commandes docker pull et docker images ou docker image ls). **Cr2_2** Un conteneur est créé à partir d'une image en utilisant les options fondamentales nécessaires à son fonctionnement (commande docker run, options --name, -d et -it) **Cr2_3** Une image est nommée afin d'être publiée dans un registre (commandes docker tag, docker push) **Cr2_4** Les informations essentielles sur la structure d'une image ou d'un conteneur telles que les volumes, la configuration système et la commande de démarrage, sont obtenues à l'aide de la commande docker inspect.

		Cr2_5 Les commandes du cycle de vie d'un conteneur telles que docker stop, docker start, docker pause, docker unpause, docker kill et docker rm sont utilisées afin de gérer les conteneurs en production. **Cr2_6** L'état de fonctionnement, les fichiers journaux et la liste des processus d'un conteneur sont obtenus en utilisant les commandes docker ps, docker logs et docker top. **Cr2_7** Le stockage des différentes couches d'une image est analysé pour comprendre les impacts sur la volumétrie et les performances des conteneurs créés à partir de cette image. **Cr2_8** Des fichiers sont copiés entre l'hôte Docker et un conteneur et vice-versa (commande docker cp) **Cr2_9** Des opérations de sauvegarde et de restauration d'images et conteneurs sont réalisées sur un hôte Docker (commandes docker export, docker import, docker save, docker load) **Cr2_10** Une image est créée à partir d'un conteneur préparé pour capturer une image (commande docker commit)
C3 - Concevoir et construire des images personnalisées en créant des Dockerfile et en générant les images avec la ligne de commandes pour déployer des conteneurs applicatifs répondant à des exigences métiers.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr3_1** Les éléments fondamentaux de la syntaxe d'un fichier Dockerfile sont mis en œuvre (instructions FROM, RUN, COPY, ADD) **Cr3_2** Des fichiers sont ajoutés dans une image pendant son processus de construction (instruction COPY avec options --chmod et --chown, instruction ADD) **Cr3_3** L'image décrite dans un Dockerfile expose correctement ses ressources (instructions EXPOSE, VOLUME, ENV, WORKDIR, USER) **Cr3_4** Le processus principal du conteneur créé à partir d'une image personnalisée est identifié dans le Dockerfile

		(instructions ENTRYPOIINT et CMD, script de lancement au format shell Bash) **Cr3_5** Une image est générée à partir d'un Dockerfile (commande docker build avec option -t, BuiltKit)
C4 - Chaîner des conteneurs avec Compose en les décrivant dans un fichier compose.yaml pour déployer des environnements applicatifs complets incluant des conteneurs, des réseaux et des volumes de stockage.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr4_1** La description d'une infrastructure complète à base de conteneurs est faite dans un fichier Compose nommé compose.yaml ou compose.yml contenant les propriétés services, networks et volumes. **Cr4_2** Un service est correctement décrit dans le fichier Compose avec l'ajout d'une nouvelle clé dans la propriété services **Cr4_3** Un réseau est correctement décrit dans le fichier Compose avec l'ajout d'une nouvelle clé dans la propriété networks **Cr4_4** Un volume est correctement décrit dans le fichier Compose avec l'ajout d'une nouvelle clé dans la propriété volumes **Cr4_5** Un service est déclaré comme dépendant d'un autre pour son démarrage (propriété depends_on) **Cr4_6** Le cycle de vie d'une infrastructure Compose est correctement géré avec la commande docker compose. **Cr4_7** Les informations sur le fonctionnement d'infrastructure Compose sont obtenues avec la commande docker compose.
C5 - Configurer des réseaux et des volumes en ligne de commande, dans les Dockerfile et dans les fichiers compose.yaml pour fiabiliser et organiser l'échange et le stockage des données des conteneurs applicatifs.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr5_1** Un réseau est créé en utilisant le driver approprié et le plan d'adressage indiqué (commande docker network create) **Cr5_2** Les informations à propos des réseaux existants sont obtenues (commandes docker network ls, docker network inspect)

		Cr5_3 Un conteneur est créé et démarré en étant attaché à un ou plusieurs réseaux (commande docker run option --network) **Cr5_4** Un conteneur est attaché et détaché dynamiquement d'un réseau (commandes docker network connect, docker network disconnect) **Cr5_5** Le port réseau d'un conteneur est mappé sur un port de l'hôte Docker (commande docker run option -p) **Cr5_6** Un volume dont l'emplacement de stockage est géré par le démon Docker est créé (commande docker volume create) **Cr5_7** Les informations à propos des réseaux existants sont obtenues (commandes docker volume ls, docker volume inspect) **Cr5_8** Un conteneur est créé et démarré et monté sur un ou plusieurs volumes (commande docker run option -v) **Cr5_9** Un volume est monté pour un conteneur.
C6 - Orchestrer des conteneurs en cluster avec Docker Swarm pour assurer la disponibilité, la fiabilité et de bonnes performances des conteneurs sur ses environnements applicatifs.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr6_1** Un cluster Docker Swarm est créé et plusieurs nœuds en sont membres (commandes docker swarm init, docker swarm join-token, docker swarm join, docker swarm leave) **Cr6_2** Des informations sur les différents nœuds du cluster sont affichées (commandes docker node ls, docker node promote, docker node demote, docker note ps, docker node inspect) **Cr6_3** Des services (ensembles de conteneurs en cluster) sont créés et déployés dans le cluster Swarm (commandes docker service create, docker service scale, docker service update) **Cr6_4** Plusieurs services liés, des réseaux et des volumes sont déployés ensemble dans un cluster Swarm à l'aide d'un fichier Compose nommé docker-stack.yml

C7 - Implémenter une stratégie de sécurité efficace en sécurisant le démon Docker et en appliquant des bonnes pratiques de conception d'architecture pour fiabiliser le stockage et l'échange de données par les conteneurs, dans un environnement basé sur des conteneurs applicatifs.	1 - Cas pratique : exécution de commandes sur un terminal ou une invite de commandes 2 - Questionnaire de type QCM/QCU à visée professionnelle	**Cr7_1** La communication entre le démon Docker et les clients est sécurisée à l'aide de clés de chiffrement **Cr7_2** Les images Docker personnalisées ainsi que les conteneurs mettent en œuvre les bonnes pratiques de sécurité telles que l'exécution des conteneurs avec un compte sans privilèges ou la mise en œuvre des espaces de nommage. **Cr7_3** Les images Docker personnalisées sont signées numériquement. **Cr7_4** Les données sensibles échangées dans un cluster Swarm sont chiffrées (commande docker secret)