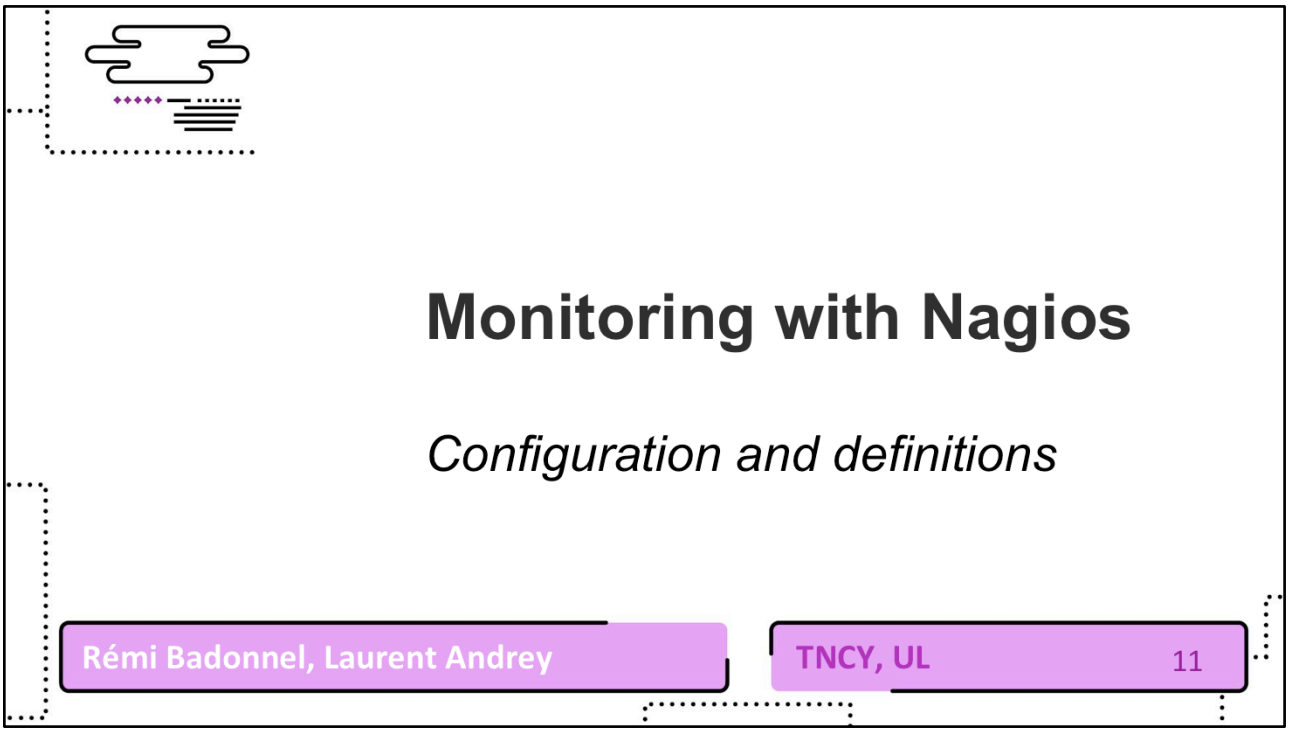
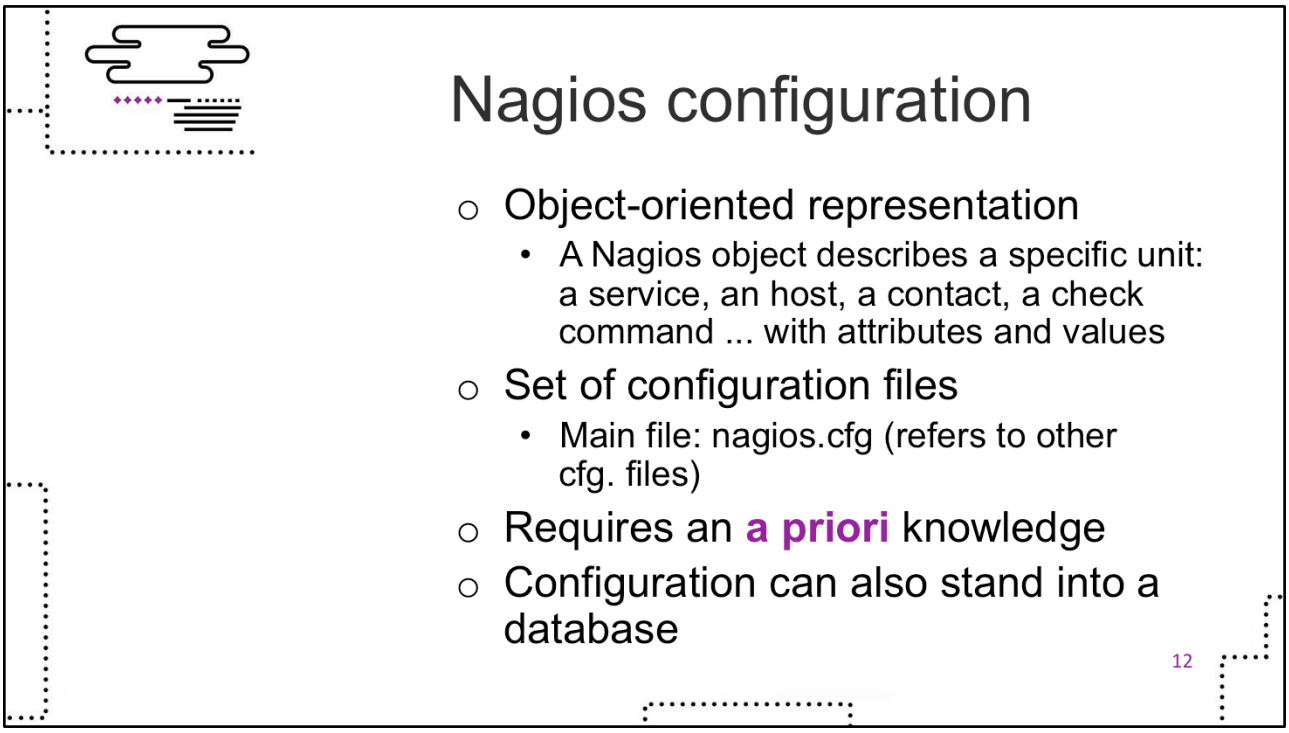


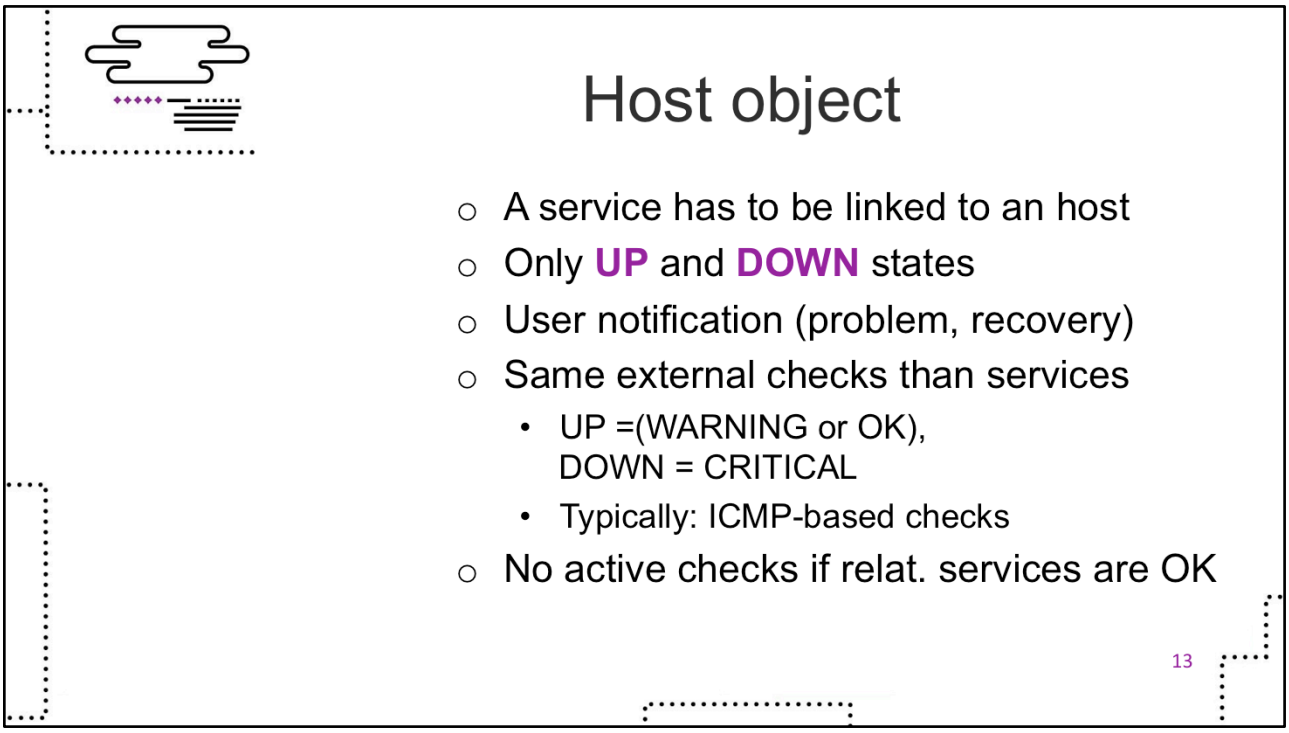
La leçon suivante va porter sur la configuration de Nagios, et en particulier, sur les différentes définitions qui vont permettre de décrire ce que l'on souhaite observer et tester sur l'infrastructure réseau.



La configuration de Nagios s'appuie sur une représentation que l'on peut qualifier d'orientée objet. Un objet Nagios va permettre de décrire une unité spécifique, tel qu'un service, un host, un contact ou un groupe de contacts, voire une commande exécutant un test. Chaque description repose sur des attributs et des valeurs associées à ces attributs. Nagios inclut également une forme d'héritage, utilisant le concept de templates.

Il est également possible de spécifier des dépendances entre les objets qui sont définis. La configuration repose typiquement sur un ensemble de fichiers. Le fichier principal nagios.cfg va faire référence aux autres fichiers de configuration. Dans une configuration classique, nous pouvons trouver un fichier de configuration par type d'objets. Par exemple, on peut avoir un fichier services.cfg qui va regrouper l'ensemble des définitions correspondant aux services.

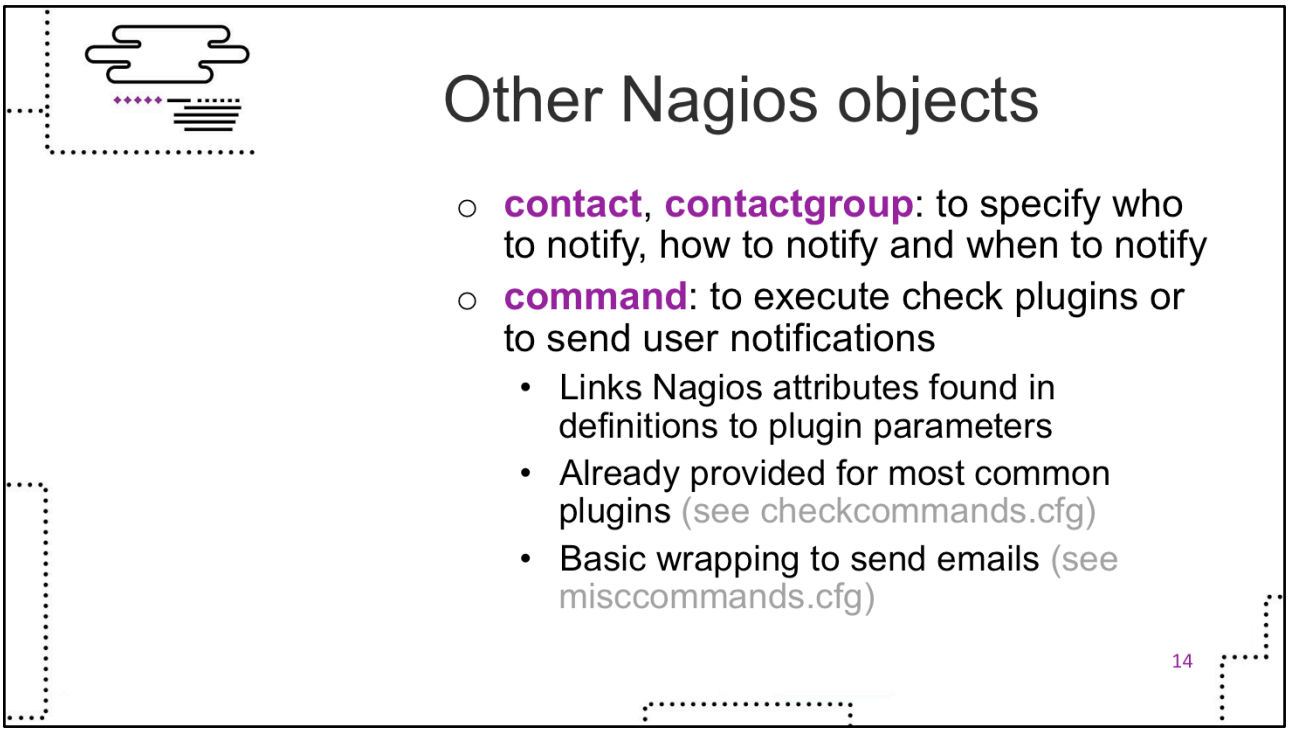
A noter que Nagios requiert une connaissance a priori, c'est-à-dire qu'il n'inclut pas de mécanismes de découverte. Il est donc nécessaire de spécifier tous les hôtes et les services que l'on souhaite surveiller dans l'infrastructure. Il est cependant possible de coupler l'outil à des outils de découverte et également de stocker les éléments de configuration dans une base de données dédiée.



Il existe différents types d'objets que l'on peut configurer avec Nagios.

En plus des services, vous pouvez configurer des objets de type host. Chaque service est en fait lié à un host. Il n'existe que deux états principaux pour les machines : un état UP et un état DOWN. Les tests sont similaires à ceux utilisés pour les services, excepté que les états WARNING et OK sont équivalents à un unique état UP, et que l' état CRITICAL correspond à l'état DOWN.

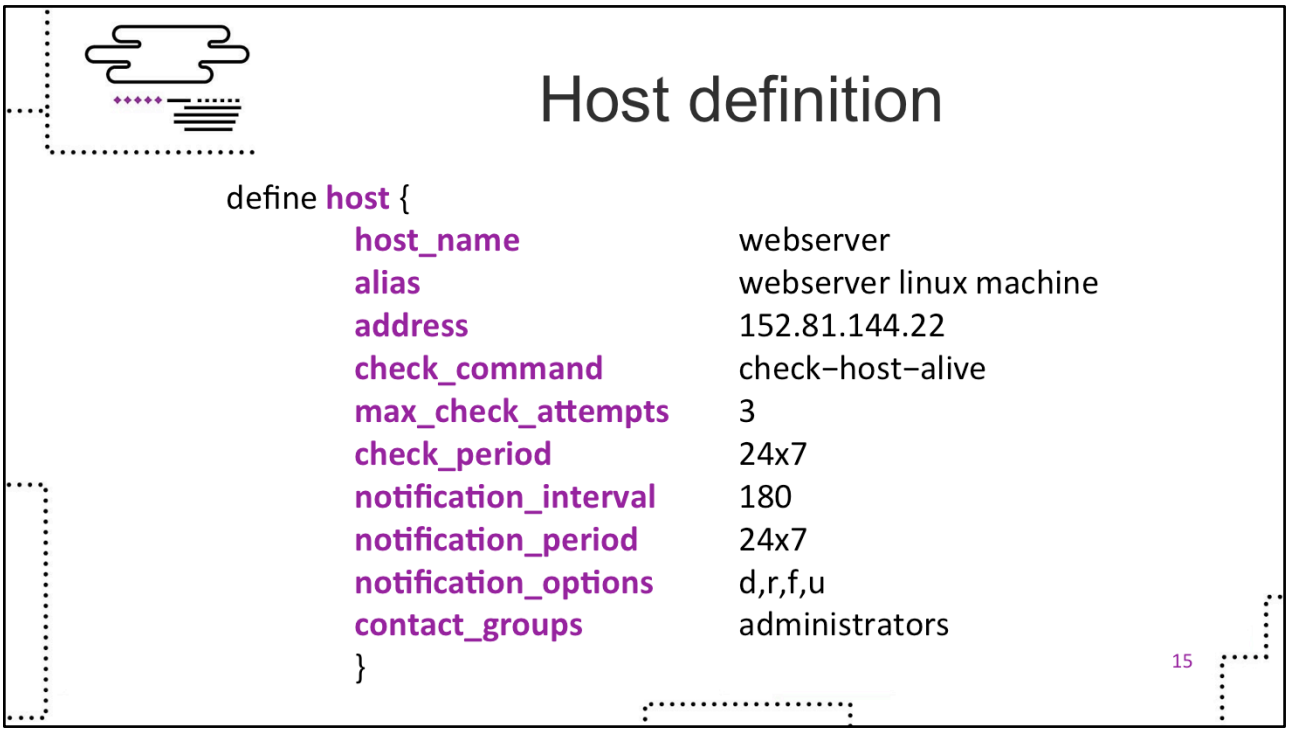
Les tests s'appuient typiquement sur des pings, en utilisant donc le protocole ICMP. Il n'y a pas de tests actifs sur un hôte si les services sont OK. Il y a ainsi une dépendance implicite, à savoir que : si un service déployé sur un hôte est opérationnel, alors cet hôte est considéré comme opérationnel.



D'autres objets importants sont présentés ici, en particulier les objets contact et contactgroup, qui permettent de spécifier à qui, comment, à quelle fréquence et sur quelle période de temps les notifications doivent être transmises aux administrateurs.

L'objet command est, quant à lui, utilisé pour exécuter les plugins de tests, et envoyer les notifications. Ces objets permettent d'utiliser les valeurs des attributs des objets Nagios pour le paramétrage des plugins. Des objets command sont déjà fournis pour la plupart des plugins et sont disponibles, typiquement dans le fichier de configuration checkcommands.cfg.

On trouve également des objets command pour l'envoi des notifications, typiquement dans le fichier de configuration misccommands.cfg.



Supposons maintenant un scénario où l'on souhaite configurer Nagios pour surveiller un serveur web. Cela signifie que nous souhaitons vérifier l'état du serveur et s'assurer qu'il fonctionne correctement.

Nous devons tout d'abord configurer Nagios pour ajouter un objet de type host, dont la définition est fournie ici. Comme vous pouvez l'observer, la définition est composée d'un ensemble d'attributs présentés sur la partie gauche de la définition en violet, et vous avez également un ensemble de valeurs qui sont associées à ces attributs et qui sont présentées en noir sur la partie droite.

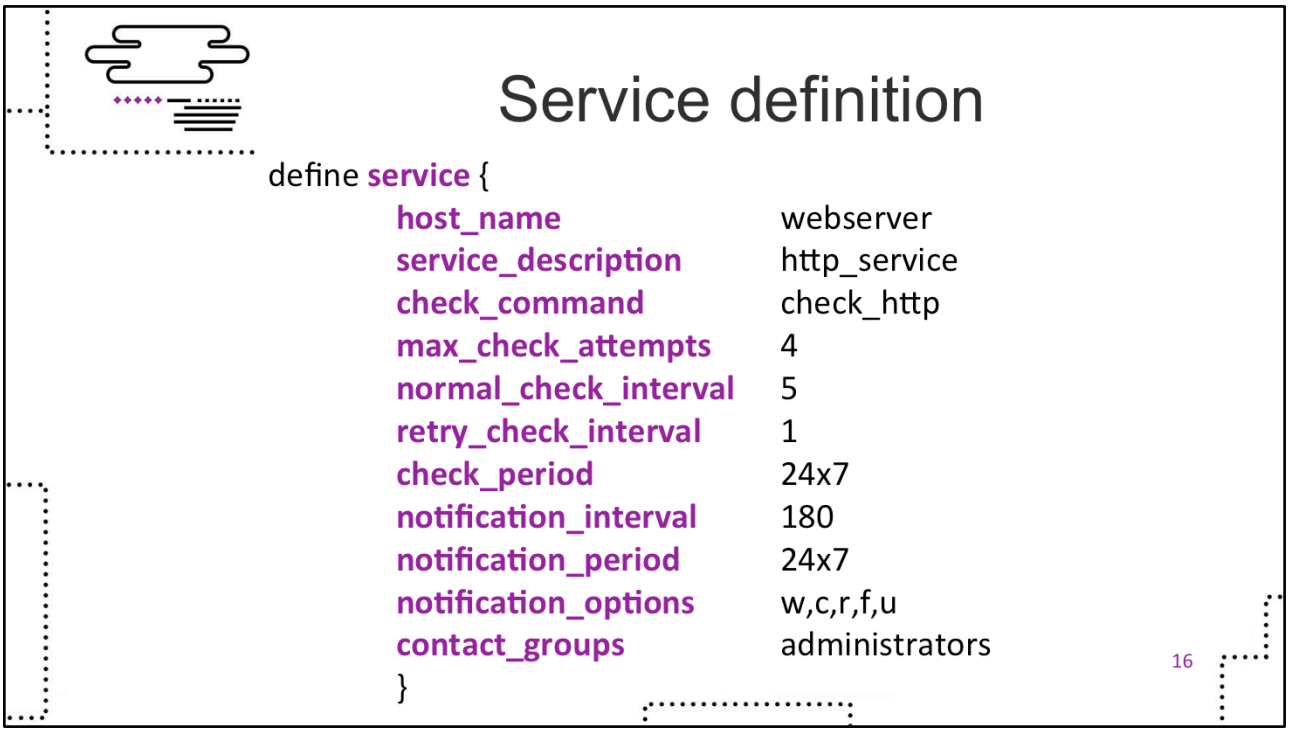
Le premier attribut ici qu'on peut constater sur la définition est host_name. Il va permettre de spécifier le nom de la machine (dans notre cas webserver). Le second attribut est appelé alias et va permettre de fournir une description longue du serveur. L'attribut suivant est l'adresse : nous avons spécifié l'adresse IP du serveur, à savoir 152.81.144.22.

L'attribut suivant check_command permet d'indiquer comment la machine va être testée. En fait, la valeur de l'attribut qui est ici check-host-alive est une référence à un objet de type command. Cette commande va permettre de tester l'état du serveur au travers de l'envoi de pings. L'attribut max_check_attempts va permettre, comme on l'a vu précédemment, d'indiquer le nombre de fois qu'un test doit être réalisé avant l'envoi de notifications. Cela signifie : avant qu'un nouvel état, l'état DOWN, soit confirmé.

L'attribut check_period permet de spécifier des périodes de temps durant lesquelles les tests sont exécutés. Ici, la valeur de l'attribut est à nouveau une référence à un autre objet, qui est un objet de type timeperiod. Cet objet va simplement décrire que la période de temps correspond à une surveillance 24 heures sur 24, 7 jours sur 7.

Les attributs suivants, notification_interval et notification_period, vont permettre de spécifier l'intervalle entre deux notifications. Ici, la valeur est de 180 minutes, soit trois heures entre deux notifications, et la période de notification, quant à elle, fait référence à nouveau au même objet timeperiod que précédemment, à savoir 24x7 correspondant à une période de notifications qui est permanente : 24 heures sur 24, 7 jours sur 7.

L'attribut notification_options permet de réaliser un filtrage sur les notifications. Vous avez en fait une liste avec les différents types de notifications que vous souhaitez récupérer. Ici, vous avez plusieurs lettres, donc d,r,f,u : d indique une notification correspondant à un état DOWN (lorsque le serveur tombe en panne), r correspond à recovery, c'est simplement pour indiquer lorsque le serveur est à nouveau opérationnel (dans un état UP), f permet d'identifier des états qui sont problématiques, lorsque vous allez avoir un serveur qui va passer régulièrement d'un état UP à un état DOWN, et enfin vous avez un dernier état u (pour UNREACHABLE) qui indique en fait des situations où Nagios n'est pas capable d'évaluer l'état du serveur. Cela arrive lorsque Nagios n'est pas capable d'atteindre le serveur, parce, par exemple, le routeur qui permet de joindre ce serveur n'est pas opérationnel. On a enfin un dernier attribut contact_groups, qui va permettre d'indiquer le groupe d'administrateurs à joindre lorsqu'une panne est détectée.



Dès lors que l'objet de type host est défini, nous allons pouvoir spécifier un objet de type service.

Le premier attribut d'un service est host_name. En fait, il s'agit d'une référence au serveur sur lequel le service fonctionne. Dans notre cas, cela va donc être web_server. Cela nous permet de lier la définition du service à la définition de la machine qu'on a présentée juste avant.

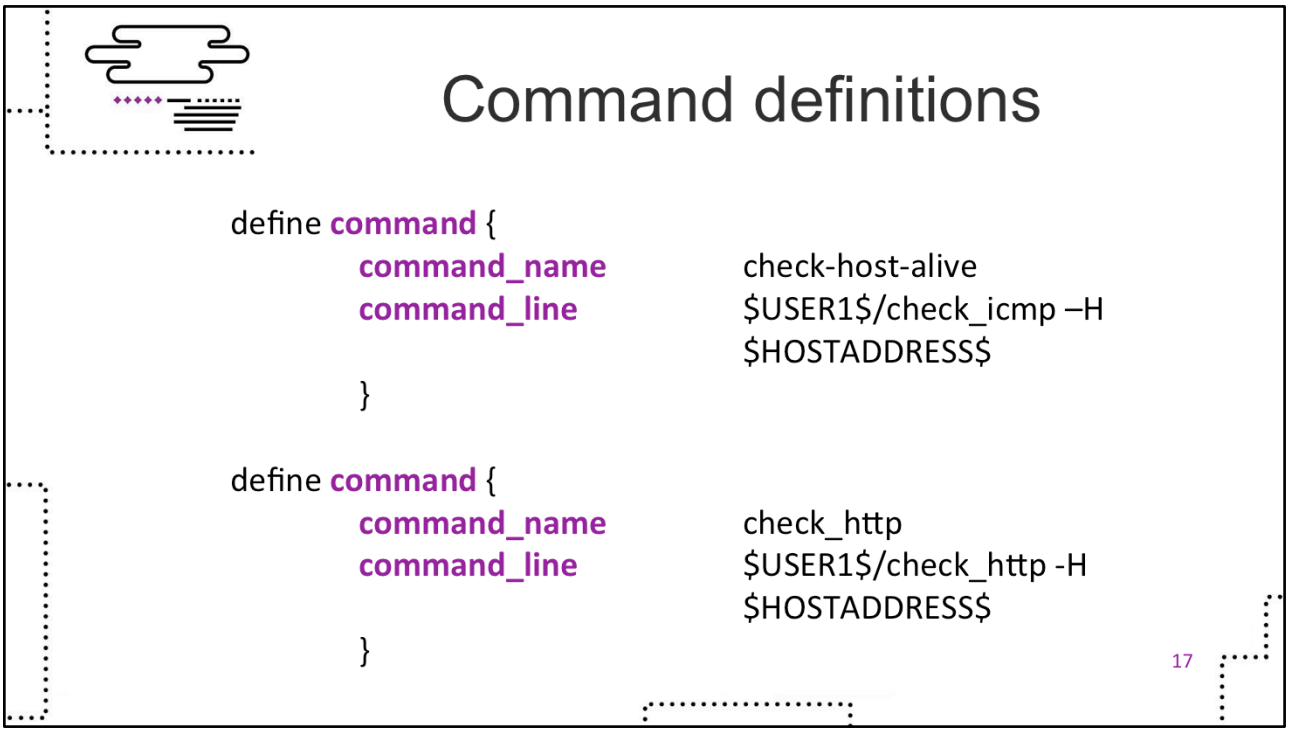
Le deuxième attribut est intitulé service_description. Il s'agit en fait plus que d'une description d'un service. Cela va être réellement un identifiant pour le service, et qui pourra être réutilisé dans d'autres définitions. Donc dans notre cas, on a donné comme valeur http_service.

L'attribut suivant est check_command. On le retrouve à nouveau ici. L'objectif est de spécifier comment l'état du service est vérifié. check_http est une référence à un objet command, et cette commande permet de vérifier l'état du service HTTP. Nous avons ensuite l'attribut max_check_attempts qui spécifie le nombre de fois que le test doit être exécuté, avant qu'un nouvel état ne soit confirmé.

On a également normal_check_interval et retry_check_interval qui indiquent la fréquence des tests. normal_check_interval permet de spécifier l'intervalle de temps entre deux tests, lorsque les choses se passent bien, à savoir lorsque le service fonctionne correctement. Donc ici, les tests vont être réalisés chaque cinq minutes, et lorsqu'un état CRITICAL ou WARNING est détecté, la fréquence va changer. Elle passe à une minute entre deux tests. L'objectif est de pouvoir rapidement confirmer l'état, c'est-à-dire passer de l'état SOFT à l'état HARD correspondant.

On a ensuite check_period qui est à nouveau ici de 24x7, référence à un objet time_period. L'intervalle de notifications (notification_interval) est le même que pour la définition qu'on a utilisée pour l'host, à savoir 180 minutes. La période de notifications est à nouveau ici fixée sur sur la même période, à savoir 24x7.

Pour les options de notifications, ça change ici un peu par rapport à la définition des hosts. On a ici plus d'états possibles pour un service, à savoir w pour les notifications correspondant aux WARNING, c pour les notifications d'états CRITICAL, r pour le rétablissement (recovery) du service, f pour flapping, et u pour unreachable. A nouveau, vous pouvez à la fin spécifier avec l'attribut contact_groups la liste des administrateurs à contacter pour ce service.



Pour terminer, il est intéressant également de regarder les objets de type command, qui sont référencés par nos définitions de type host et service.

On a ici deux définitions de type command. La première qui correspond à la commande check-host-alive. Elle est utilisée dans la définition de web_server. L'attribut command_line va permettre d'indiquer explicitement la ligne de commandes qui va lancer le plugin correspondant. Vous avez quelques macros qui sont utilisés. Vous avez $USER1$ qui va permettre d'indiquer le répertoire où se trouve le plugin, et vous avez $HOSTADDRESS$ qui va permettre de récupérer l'adresse IP de l'host. Cela va récupérer directement la valeur de l'attribut correspondant dans la définition de la machine. Nous avons la même chose pour la seconde définition. Dans ce cas, le nom de la commande est check_http, et on trouve ensuite la ligne de commandes qui s'appuie à nouveau sur les mêmes macros pour appeler le plugin check_http.

L'objectif de ces définitions de type command est clairement de lier les paramètres des objets qui ont été définis avec Nagios et les plugins qui seront ensuite exécutés.