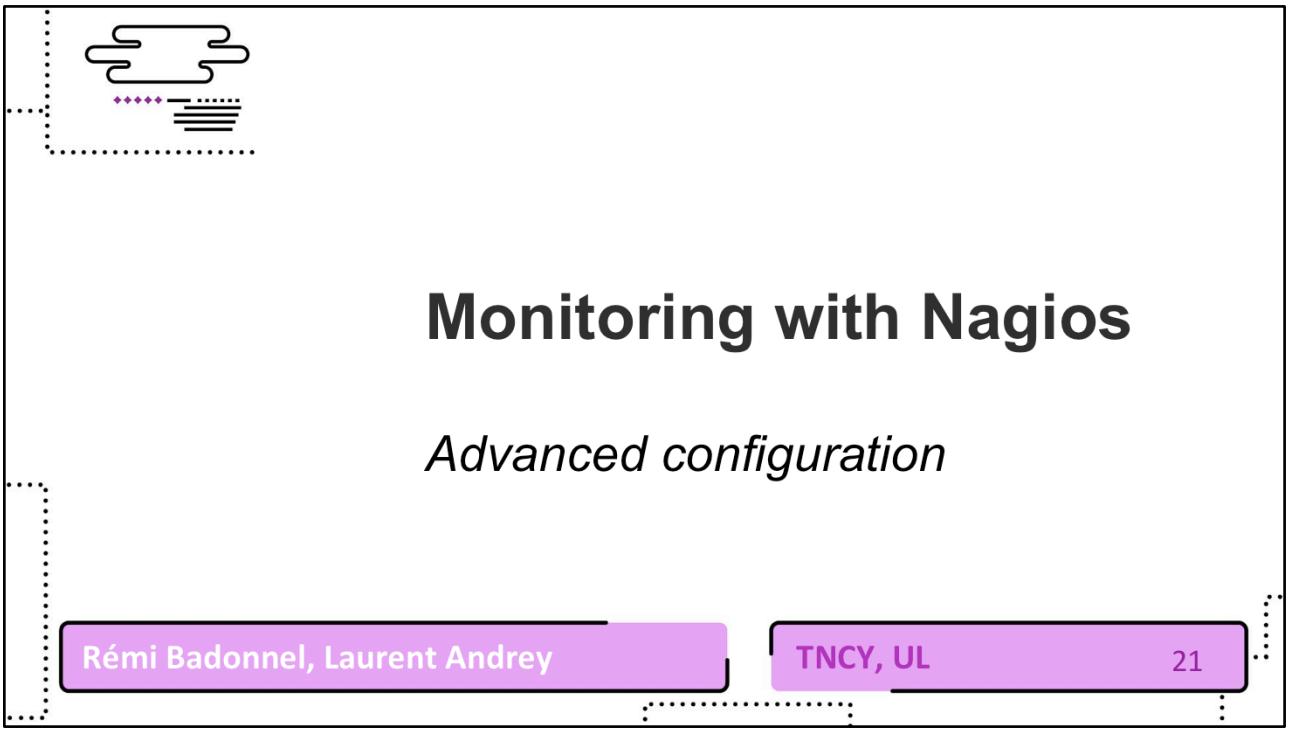
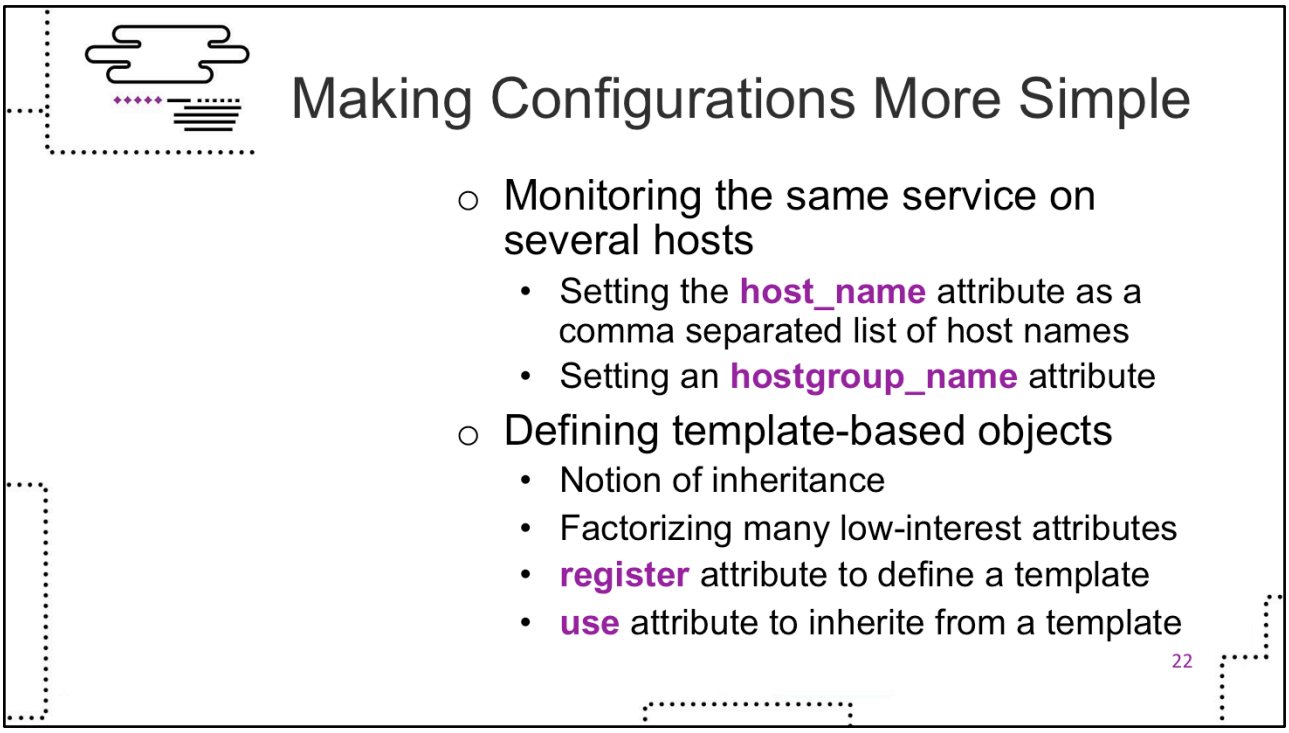


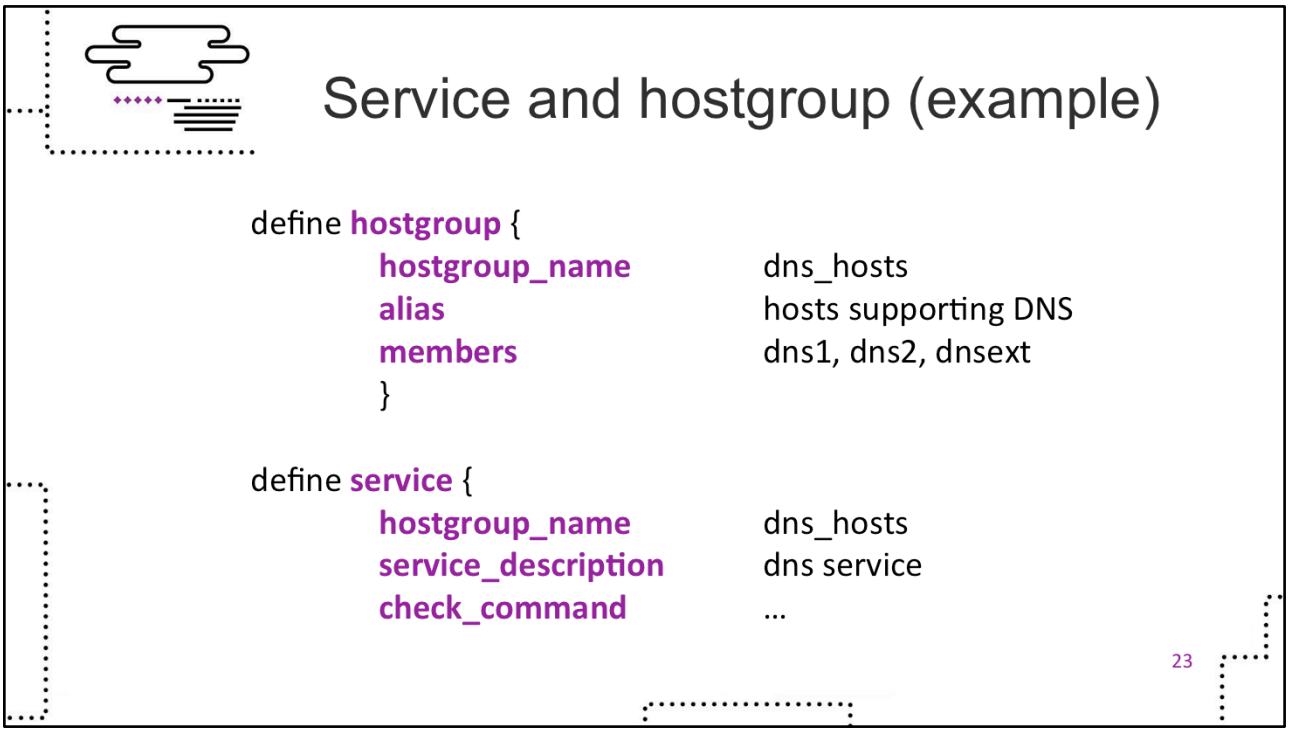
Dans cette dernière leçon, nous allons traiter de méthodes de configuration avancée, avec l'outil Nagios.



Une première méthode pour améliorer la configuration sous Nagios est de rendre les configurations plus simples. Vous avez deux façons pour le faire.

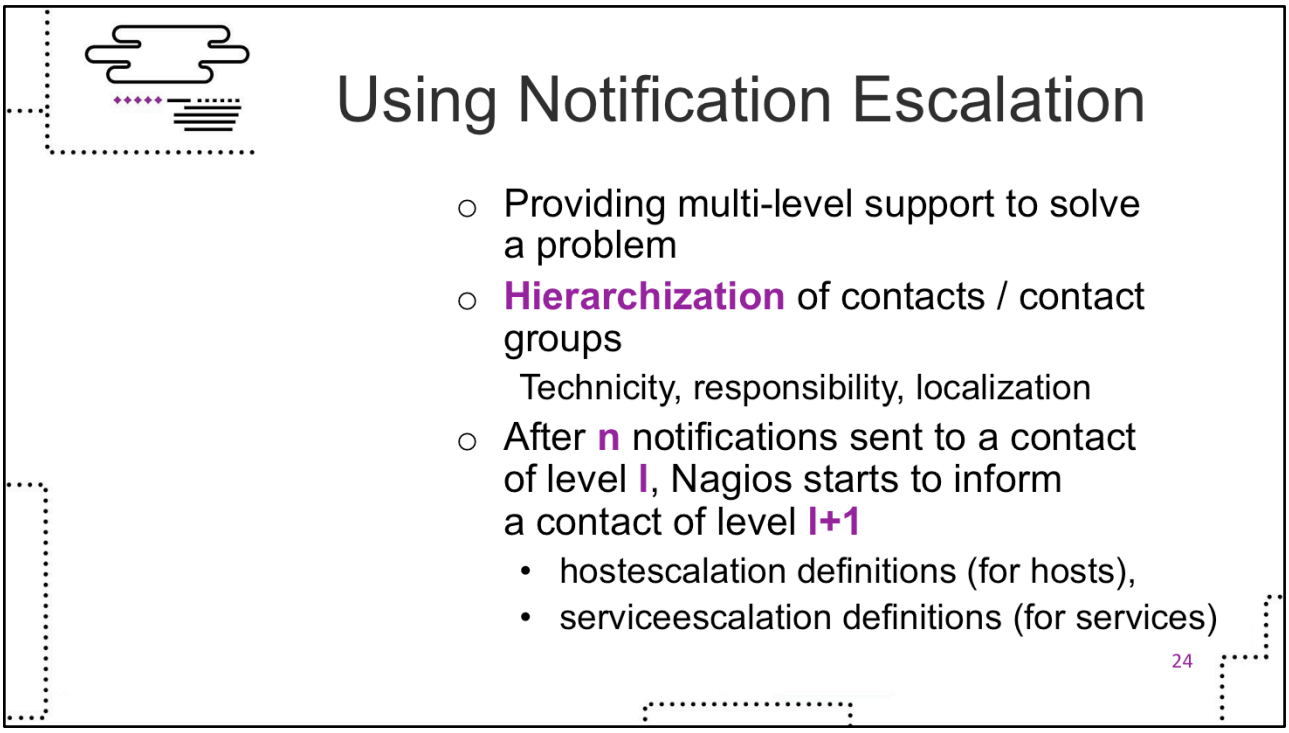
La première méthode est possible lorsque vous surveillez plusieurs fois le même service sur des machines différentes. Il va suffir de définir un unique objet de type service et de le faire référencer plusieurs noms de machines. Pour ce faire, il faut spécifier comme valeur pour l'attribut host_name une liste de plusieurs noms d'hôtes séparés par des virgules. Une autre solution consiste à utiliser un autre attribut de type hostgroup_name au lieu de l'attribut host_name. L'objet hostgroup qui sera fourni comme valeur à cet attribut fera lui-même référence à la liste de machines.

Une seconde façon de simplifier la configuration est d'utiliser les templates. Nagios fournit ainsi des mécanismes qui sont proches de l'héritage, et qui vont permettre de factoriser au maximum les attributs de moindre intérêt. Pour définir un template, il suffit d'ajouter un attribut register avec une valeur de 0 à une définition classique, par exemple la définition d'un hôte ou d'un service. Dès lors, la définition n'est plus considérée comme celle d'un objet à surveiller, mais comme un template qui va pouvoir être réutilisé pour définir d'autres définitions. La réutilisation des templates est alors possible grâce à l'ajout d'un attribut use à une définition donnée. La valeur de l'attribut doit correspondre au nom du template. En conséquence, tous les attributs définis dans le template seront obtenus par héritage dans cette nouvelle définition.



Cette planche illustre l'utilisation d'un objet de type hostgroup pour surveiller les services fournis par trois serveurs DNS. Au lieu de définir un service pour chacun de ces serveurs, l'approche consiste à introduire un objet de type hostgroup. Dans notre exemple, celui-ci est appelé dns_hosts, et indique un ensemble de machines grâce à l'attribut members, à savoir les machines : dns1, dns2, et dnsext.

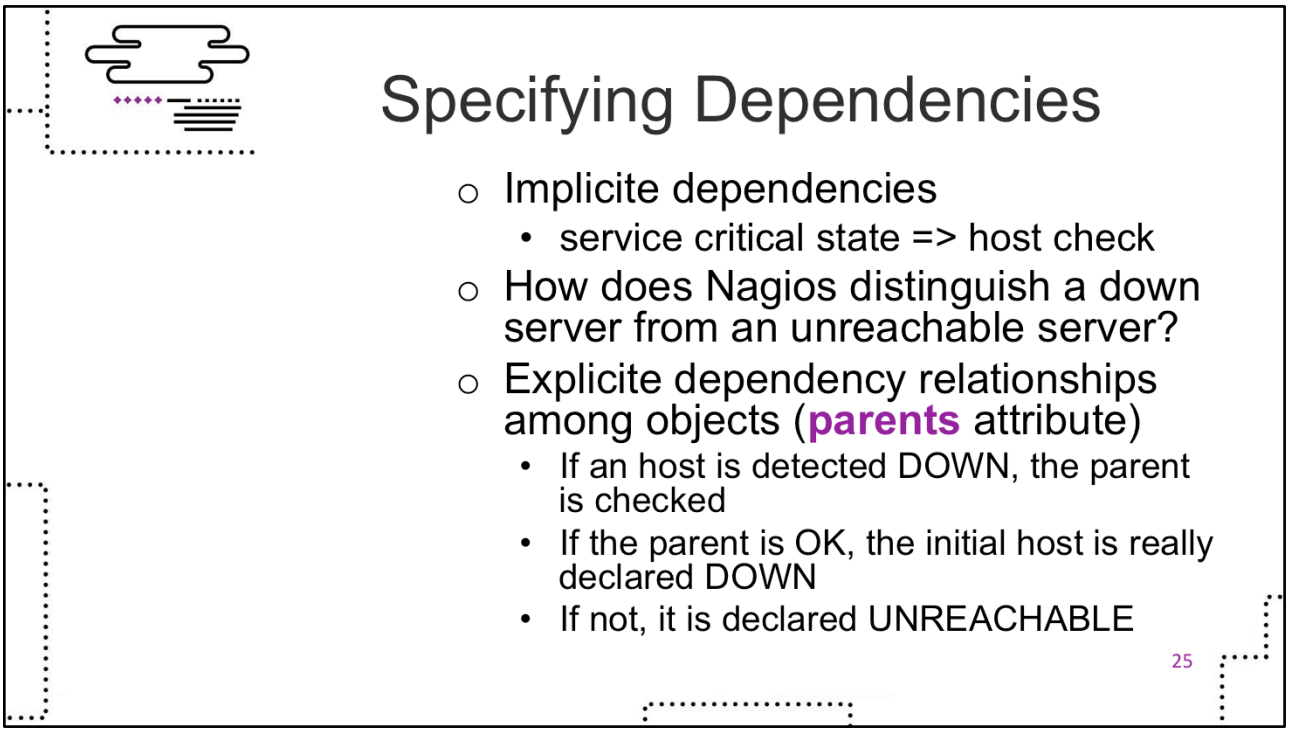
Dans la définition du service, l'attribut host_name est alors remplacé par un attribut hostgroup_name, celui-ci faisant référence à dns_hosts. Dans ce cas, une seule définition de type service est nécessaire pour surveiller chacun des services fournis par les serveurs DNS. Ceci permettant de réduire le nombre de définitions dans la configuration Nagios, et par là même de minimiser les erreurs potentielles.



La force de Nagios vient également de son système de notifications.

Pour les configurations avancées, un concept important est celui d'escalade de notifications. L'objectif est de fournir un support multi-niveaux pour la résolution de problèmes, en s'appuyant sur une hiérarchie des contacts et groupes de contacts. Cette hiérarchie peut reposer sur le niveau de technicité, le niveau de responsabilité, ou plus simplement la localisation géographique des contacts.

L'escalade de notifications consiste à envoyer tout d'abord des notifications à un contact de niveau donné, le niveau que l'on peut noter l. Si après n notifications, le problème n'est toujours pas résolu, alors Nagios va commencer à informer le contact de niveau l+1. Si nous prenons l'exemple d'une hiérarchie fondée sur le niveau de technicité, Nagios peut tout d'abord contacter un technicien du front office, avant d'informer celui du back office après un certain nombre de notifications. La configuration de ce type de notifications requiert l'ajout de définitions supplémentaires, qui correspondent aux définitions host_escalation pour les machines, et aux définitions service_escalation pour les services.

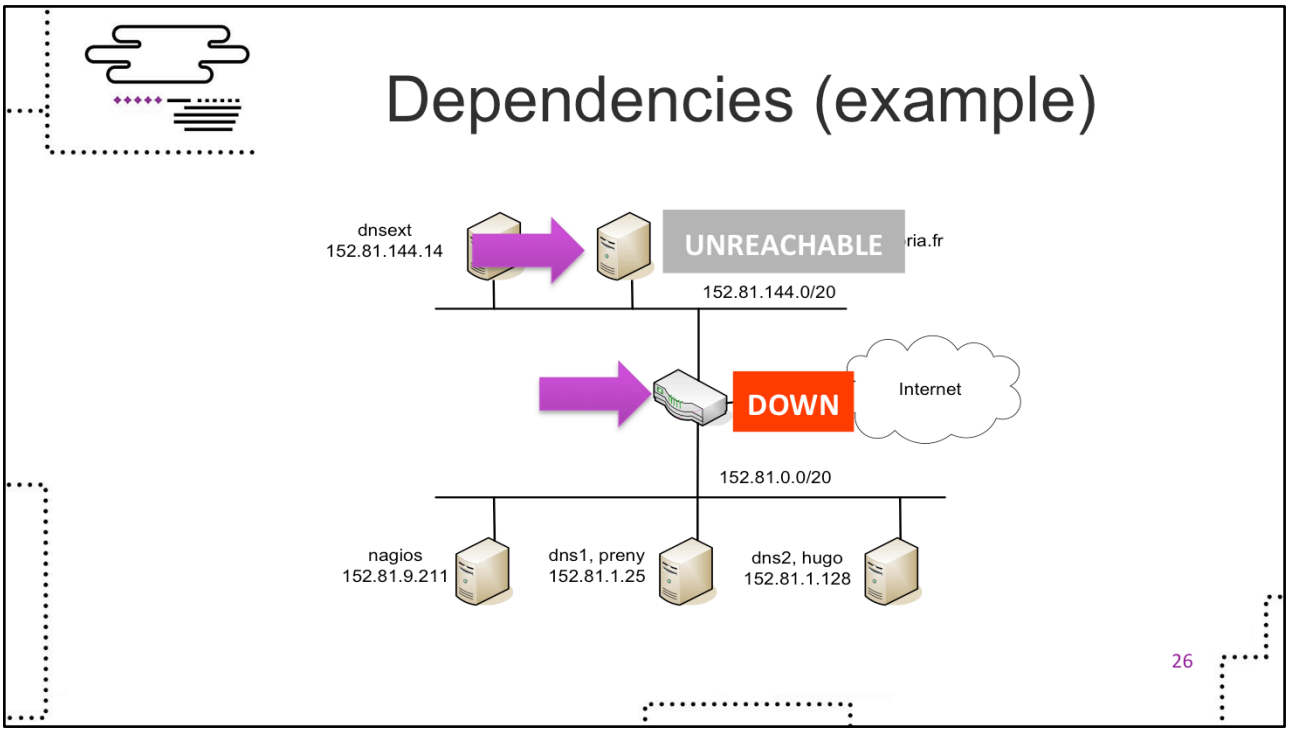


Nagios vise à détecter les pannes, et ce le plus tôt possible. Pour les configurations avancées, il est possible de spécifier des dépendances pour améliorer l'analyse des pannes.

Nagios inclut déjà des dépendances implicites entre les services et les hôtes. Un hôte n'est pas testé directement, si ses services sont opérationnels.

Nous pouvons également ajouter des dépendances explicites. En particulier, un problème typique peut se poser, lorsque l'on souhaite distinguer un serveur en panne d'un serveur qui est simplement inatteignable parce que le routeur permettant de le joindre est inopérant. Une solution consiste alors à introduire une relation de dépendance entre les objets Nagios, en utilisant l'attribut parents.

Dans ce cas, si une machine est détectée comme en panne, alors l'objet parent va être testé. Si le parent fonctionne, alors la machine initiale est déclarée comme étant effectivement en panne. Si non, la machine est simplement considérée comme étant inatteignable.

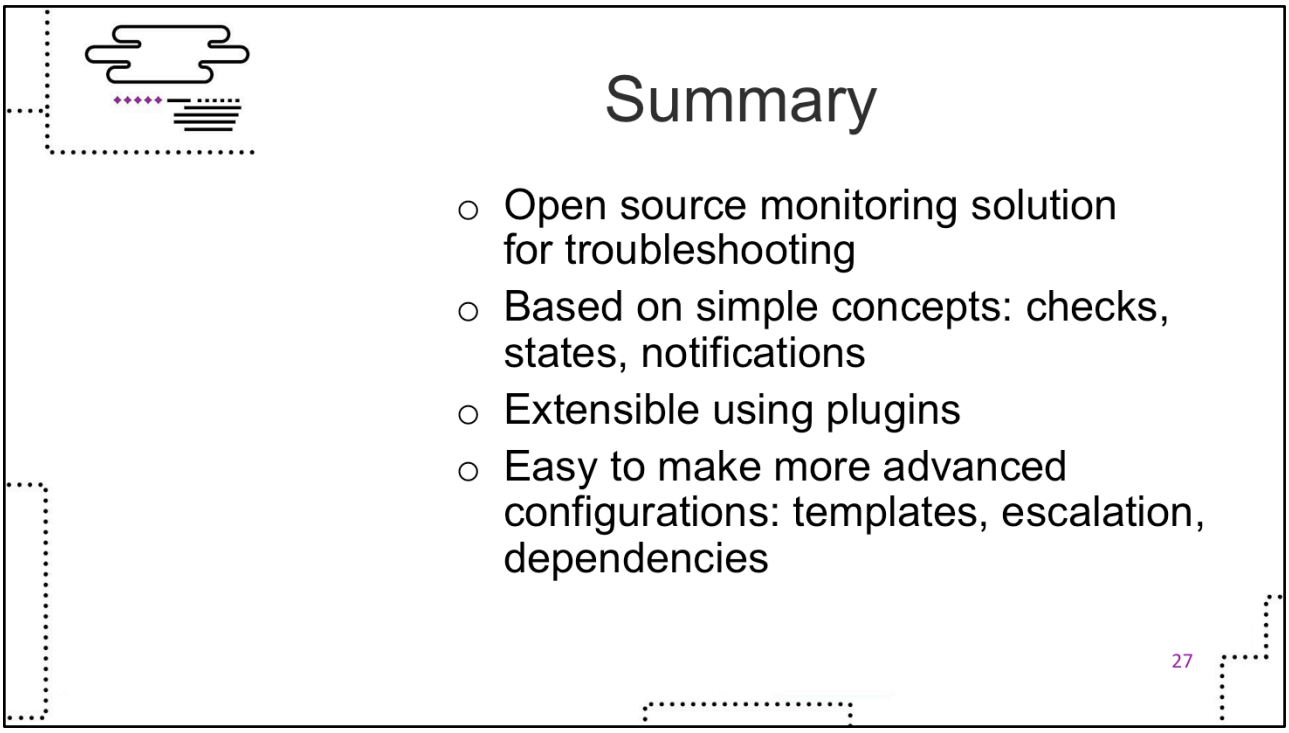


Prenons un exemple simple pour clarifier ce concept. Supposons que l'on dispose de l'infrastructure suivante : elle est composée de deux sous-réseaux séparés par un routeur. Un premier sous-réseau sur lequel se trouve le serveur web que nous souhaitons surveiller et noté webloria, et un second sur lequel se trouve le serveur Nagios. Nagios teste régulièrement l'état du serveur web. Mais comment peut-il distinguer la panne du serveur web de celle du routeur qui se trouve entre les deux sous-réseaux ?

Nous allons pour ce faire déclarer le routeur comme parent du serveur web. Si le serveur web est détecté comme en panne, alors Nagios va vérifier l'état du routeur. Si le routeur et dans l'état OK, Nagios va considérer que le serveur web est effectivement en panne, donc dans l'état DOWN.

Si ce n'est pas le cas, à savoir si le routeur est détecté comme DOWN, alors Nagios va simplement considérer le serveur web comme non atteignable, donc dans l'état UNREACHABLE.

La spécification de telles dépendances va donc permettre d'améliorer la détection des pannes.



Pour conclure, Nagios fournit une solution de monitoring, qui est open source, largement déployée, qui va permettre la détection de pannes et de dysfonctionnements au sein d'une infrastructure réseau.

Nagios s'appuie sur des concepts simples tels que ceux liés aux tests, aux états et aux notifications.

L'outil est facilement extensible à travers le développement de nouveaux plugins, qui peuvent être de simples scripts et qui permettent de répondre à des besoins spécifiques.

Enfin, il est également possible de construire des configurations plus élaborées, en exploitant les facilités offertes par les templates, l'escalade de notifications, et la spécification de dépendances.