Nagios Admin Interview Questions

* **What Are Plugins In Nagios?**

Plugins are scripts (Perl scripts, Shell scripts, etc.) that can run from a command line to check the status of a host or service. Nagios uses the results from the plugins to determine the current status of hosts and services on your network.

Once you have defined Plugins I will suggest you to explain why we need plugins.

Nagios will execute a Plugin whenever there is a need to check the status of a host or service. The plugin will perform the check and then simply returns the result to Nagios. Nagios will process the results that it receives from the Plugin and take the necessary actions.

* **What Is Nagios And How It Works?**

Nagios is an open source System and Network Monitoring application. Nagios runs on a server, usually as a daemon or service. Nagios periodically run plugins to monitor clients, if it found anything warning and critical it will send an alerts via Email OR SMS as per the configuration.

The Nagios daemon behaves like a scheduler that runs certain scripts at certain moments. It stores the results of those scripts and will run other scripts if these results change.

* **What Are Ports Numbers Nagios Will Use To Monitor Clients?**

Port numbers are 5666, 5667 and 5668

* **Explain Main Configuration File And Its Location?**
  1. **Resource File** : It is used to store sensitive information like username, passwords with out making them available to the CGIs. Default path: /usr/local/nagios/etc/resource.cfg
  2. **Object Definition Files:** It is the location were you define all you want to monitor and how you want to monitor. It is used to define hosts, services, hostgroups, contacts, contact groups, commands, etc.. Default Path:/usr/local/nagios/etc/objects/
  3. **CGI Configuration File :** The CGI configuration file contains a number of directives that affect the operation of the CGIs. It also contains a reference the main configuration file, so the CGIs know how you’ve configured Nagios and where your object definitions are stored. Default Path: /usr/local/nagios/sbin/
* **Nagios Administrator Is Adding 100+ Clients In Monitoring But He Don’t Want To Add Every .cfg File Entry In Nagios.cfg File He Want To Enable A Directory Path. How Can He Configure Directory For All Configuration Files?**

He can able to achieve the above scenario by adding the directory path in nagios.cfg file, in line number 54 we have to add below line.

54  cfg\_dir=/usr/local/nagios/etc/objects/monitor

* **What Is Nagios?**

Nagios is one of the monitoring tools. It is used for Continuous monitoring of systems, applications, services, and business processes etc. in a DevOps culture. In the event of a failure, Nagios can alert technical staff of the problem, allowing them to begin remediation processes before outages affects business processes, end-users, or customers. With Nagios you don’t have to explain why an unseen infrastructure outage affect your organization’s bottom line.

* **Now, Once You Have Defined What Is Nagios, You Can Mention The Various Things That You Can Achieve Using Nagios?**

By using Nagios you can:

• Plan for infrastructure upgrades before outdated systems cause failures.

• Respond to issues at the first sign of a problem.

• Automatically fix problems when they are detected.

• Coordinate technical team responses.

• Ensure your organization’s SLAs are being met.

• Ensure IT infrastructure outages have a minimal effect on your organization’s bottom line.

• Monitor your entire infrastructure and business processes.

This overall completes the answer to this question. The further details like advantages etc. can be added as per the direction where the discussion is heading.

* **Explain Nagios State Types?**
  1. The status of service or host i.e. OK, WARNING, UP, DOWN etc..
  2. The type of state the service or host is in.
  3. There are two types of states SOFT states and HARD states.
* **Explain What Is Soft And Hard States?**
  1. When a service or host check results are in a non-OK or non-UP state and the service check has not yet been rechecked the number of times specified by the max\_check\_attempts directives in the service or host definition. This is called Soft Error. When a service or a host recovers from Soft Error that is considered as Soft Recovery.
  2. When a service or host check results are in a non-OK or non-UP state and the service check has been rechecked the number of times specified by the max\_check\_attempts directives in the service or host definition. This is called Hard Error. When a service or a host recovers from Hard Error that is considered as Hard Recovery.
* **Nagios Says My Machine Is Unreachable, Not Down. What Is The Difference And How It Is Achieved?**

When Nagios says a node is unreachable, a node is unreachable if Nagios is not able to find a path to the node.   
Now you can mention the difference.

The node itself may be up but because Nagios is unable to connect to it, it has to mark this as unreachable. To achieve this, Nagios use parent-child relationship between components.

Finally for better understanding explain it with an example.

* 1. A router may be defined as a parent for a server.
  2. Now Nagios checks for server and marks it as down.
  3. It then checks the parent (in our example, the router)
  4. If parent is also down, then server is marked as unreachable.
  5. If Parent is up, the server is marked as really down.
* **What Is State Stalking In Nagios?**

State Stalking is used for logging purposes. When Stalking is enabled for a particular host or service, Nagios will watch that host or service very carefully and log any changes it sees in the output of check results.

Depending on the discussion between you and interviewer you can also add:

It can be very helpful in later analysis of the log files. Under normal circumstances, the result of a host or service check is only logged if the host or service has changed state since it was last checked.

* **What Is Meant By Saying Nagios Is Object Oriented?**

One of the features of Nagios is object configuration format in that you can create object definitions that inherit properties from other object definitions and hence the name. This simplifies and clarifies relationships between various components.

* **What Are The Three Main Variables That Affect Recursion And Inheritance In Nagios?**

**First name the variables and then a small explanation of each of these variables:**

* 1. Name
  2. Use
  3. Register

Now I will give a small explanation for each of these variables.

Name is a placeholder that is used by other objects. Use defines the “parent” object whose properties should be used. Register can have a value of 0 (indicating its only a template) and 1 (an actual object). The register value is never inherited.

* **Explain How Flap Detection Works In Nagios?**

Flapping occurs when a service or host changes state too frequently, this causes lot of problem and recovery notifications.

Once you have defined Flapping explain how Nagios detects Flapping.

Whenever Nagios checks the status of a host or service, it will check to see if it has started or stopped flapping. Nagios follow the below procedure to do that:

* 1. Storing the results of the last 21 checks of the host or service analyzing the historical check results and determine where state changes/transitions occur.
  2. Using the state transitions to determine a percent state change value (a measure of change) for the host or service.
  3. Comparing the percent state change value against low and high flapping thresholds
  4. A host or service is determined to have started flapping when its percent state change first exceeds a high flapping threshold.
  5. A host or service is determined to have stopped flapping when its percent state goes below a low flapping threshold.
* **Explain Main Configuration File Of Nagios And Its Location?**

The main configuration file contains a number of directives that affect how the Nagios daemon operates. This config file is read by both the Nagios daemon and the CGIs (It specifies the location of your main configuration file).

Now you can tell where it is present and how it is created.

A sample main configuration file is created in the base directory of the Nagios distribution when you run the configure script. The default name of the main configuration file is nagios.cfg, it is usually placed in the etc/ subdirectory of you Nagios installation (i.e. /usr/local/nagios/etc/).

* **How Does Nagios Help With Distributed Monitoring?**

With Nagios you can monitor your whole enterprise by using a distributed monitoring scheme in which local slave instances of Nagios perform monitoring tasks and report the results back to a single master. You manage all configuration, notification, and reporting from the master, while the slaves do all the work. This design takes advantage of Nagios’s ability to utilize passive checks i.e. external applications or processes that send results back to Nagios. In a distributed configuration, these external applications are other instances of Nagios.

* **What Is The Difference Between Active And Passive Check In Nagios?**

The major difference between Active and Passive checks is that Active checks are initiated and performed by Nagios, while passive checks are performed by external applications.

**Passive checks are useful for monitoring services that are:**

* 1. Asynchronous in nature and cannot be monitored effectively by polling their status on a regularly scheduled basis.
  2. Located behind a firewall and cannot be checked actively from the monitoring host.
  3. The main features of Actives checks are as follows:
     + Active checks are initiated by the Nagios process.
     + Active checks are run on a regularly scheduled basis.
* **When Does Nagios Check For External Commands?**

**Nagios check for external commands under the following conditions:**

* 1. At regular intervals specified by the command\_check\_interval option in the main configuration file or,
  2. Immediately after event handlers are executed. This is in addition to the regular cycle of external command checks and is done to provide immediate action if an event handler submits commands to Nagios.
* **What Do You Mean By Passive Check In Nagios?**

Passive checks are initiated and performed by external applications/processes and the Passive check results are submitted to Nagios for processing.

* **Explain The Need For Passive Check?**

Passive checks are useful for monitoring services that are Asynchronous in nature and cannot be monitored effectively by polling their status on a regularly scheduled basis. It can also be used for monitoring services that are Located behind a firewall and cannot be checked actively from the monitoring host.

* **What Is Meant By Nagios Backend?(unable To Find A Relevant Explanation)**

Both Configuration and Logs can be stored in a backend. Configurations are stored in backend using NagiosQL. Historical data are stored using ndoutils. In addition, you also have nagdb and opdb.

* **What Is Database Is Used By Nagios To Store Collected Status Data?**

Nagios core will use default RRD database format to store status data

* **What Are The Operating Systems We Can Monitor Using Nagios..?**

Any Operating System We can monitor using Nagios, OS should support to install Nagios Clinet either SNMP.

* **What Are The Components That Make Up The Ndo Utilities ?**

**There are four main components that make up the NDO utilities:**

* 1. **NDOMOD Event Broker Module :** The NDO utilities includes a Nagios event broker module (NDOMOD.O) that exports data from the Nagios daemon.Once the module has been loaded by the Nagios daemon, itcan access all of the data and logic present in the running Nagios process.The NDOMOD module has been designed to export configuration data, as well as information about various run time events that occur in the monitoring process, from the Nagios daemon. The module can send this data to a standard file, a Unix domain socket, or a TCP socket.
  2. **LOG2NDO Utility :** The LOG2NDO utility has been designed to allow you to import historical Nagios and NetSaint log files into a database via the NDO2DB daemon (described later). The utility works by sending historical log file data to a standard file, a Unix domain socket, or a TCP socket in a format the NDO2DB daemon understands. The NDO2DB daemon can then be used to process that output and store the historical log file  information in a database.
  3. **FILE2SOCK Utility :**  The FILE2SOCK utility is quite simple. Its reads input from a standard file (or STDIN) and writes all of that data to either a Unix domain socket or TCP socket. The data that is read is not processed in any way before it is sent to the socket.
  4. **NDO2DB Daemon:** The NDO2DB utility is designed to take the data output from the NDOMOD and LOG2NDO components and store it in a MySQL or PostgreSQL database.When it starts, the NDO2DB daemon creates either a TCP or Unix domain socket and waits for clients to connect. NDO2DB can run either as a standalone, multi-process daemon or under INETD (if using a TCP socket). Multiple clients can connect to the NDO2DB daemon’s socket and transmit data simultaneously. A separate NDO2DB process is spawned to handle each new client that connects. Data is read from each client and stored in a user-specified database for later retrieval and processing.
* **What Is Ndoutils ?**

The NDOUTILS addon is designed to store all configuration and event data from Nagios in a database. Storing information from Nagios in a database will allow for quicker retrieval and processing of that data and will help serve as a foundation for the development of a new PHP-based web interface in Nagios 4.1.

MySQL databases are currently supported by the addon and PostgreSQL support is in development.

The NDOUTILS addon was designed to work for users who have:

* 1. Single Nagios installations
  2. Multiple standalone or “vanilla” Nagios installations
  3. Multiple Nagios installations in distributed, redundant, and/or failover environments.

Each Nagios process, whether it is a standalone monitoring server or part of a distributed, redundant, or failover monitoring setup, is referred to as an “instance”. In order to maintain the integrity of stored data, each Nagios instance must be labeled with a unique identifier or name.

* **What Is Nrpe?**

The Nagios Remote Plugin Executor addon is designed to allow you to execute Nagios plugins on remote Linux/Unix machines. The main   
reason for doing this is to allow Nagios to monitor “local” resources (like CPU load, memory usage, etc.) on remote machines. Since these public resources are not usually exposed to external machines, an agent like NRPE must be installed on the remote Linux/Unix machines.

**The NRPE addon consists of two pieces:**

* 1. The check\_nrpe plugin, which resides on the local monitoring machine
  2. The NRPE daemon, which runs on the remote Linux/Unix machine

**When Nagios needs to monitor a resource of service from a remote Linux/Unix machine:**

* 1. Nagios will execute the check\_nrpe plugin and tell it what service needs to be checked
  2. The check\_nrpe plugin contacts the NRPE daemon on the remote host over an (optionally) SSL-protected connection
  3. The NRPE daemon runs the appropriate Nagios plugin to check the service or resource
  4. The results from the service check are passed from the NRPE daemon back to the check\_nrpe plugin, which then returns the check results to the Nagios process.
* **Explain Distributed Monitoring ?**

Nagios can be configured to support distributed monitoring of network services and resources.

When setting up a distributed monitoring environment with Nagios, there are differences in the way the central and distributed servers are configured.

The function of a distributed server is to actively perform checks all the services you define for a “cluster” of hosts. it basically just mean an arbitrary group of hosts on your network. Depending on your network layout, you may have several clusters at one physical location, or each cluster may be separated by a WAN, its own firewall, etc. There is one distributed server that runs Nagios and monitors the services on the hosts in each cluster. A distributed server is usually a bare-bones installation of Nagios. It doesn’t have to have the web interface installed, send out notifications, run event handler scripts, or do anything other than execute service checks if you don’t want it to.

The purpose of the central server is to simply listen for service check results from one or more distributed servers. Even though services are occasionally actively checked from the central server, the active checks are only performed in dire circumstances.

* **Explain How Flap Detection Works In Nagios?**

Nagios supports optional detection of hosts and services that are “flapping”. Flapping occurs when a service or host changes state too frequently, resulting in a storm of problem and recovery notifications. Flapping can be indicative of configuration problems (i.e. thresholds set too low), troublesome services, or real network problems.

**Whenever Nagios checks the status of a host or service, it will check to see if it has started or stopped flapping. It does this by:**

* 1. Storing the results of the last 21 checks of the host or ser vice
  2. Analyzing the historical check results and determine where state changes/transitions occur
  3. Using the state transitions to determine a percent state change value (a measure of change) for the host or service
  4. Comparing the percent state change value against low and high flapping thresholds
  5. A host or service is determined to have started flapping when its percent state change first exceeds a high flapping threshold.
  6. A host or service is determined to have stopped flapping when its percent state goes below a low flapping threshold (assuming that is was previously flapping).
  7. The historical service check results are examined to determine where state changes/transitions occur. State changes occur when an archived state is different from the archived state that immediately precedes it chronologically. Since we keep the results of the last 21 service checks in the array, there is a possibility of having at most 20 state changes.

The flap detection logic uses the state changes to determine an overall percent state change for the service. This is a measure of volatility/change for the service. Services that never change state will have a 0% state change value, while services that change state each time they’re checked will have 100% state change. Most services will have a percent state change somewhere in between.

* **What Is State Stalking?**

Stalking is purely for logging purposes.When stalking is enabled for a particular host or service, Nagios will watch that host or service very carefully and log any changes it sees in the output of check results. As you’ll see, it can be very helpful to you in later analysis of the log files. Under normal circumstances, the result of a host or service check is only logged if the host or service has changed state since it was last checked. There are a few exceptions to this, but for the most part, that’s the rule.

If you enable stalking for one or more states of a particular host or service, Nagios will log the results of the host or service check if the output from the check differs from the output from the previous check.

* **What Is The Difference Between Nagiosxi And Nagios Core?**

NagiosXI is a Paid version and Nagios core is a free version.

NagiosXI includes lot of features which we can modify using web interface. Nagios Core default not include all the features we have to implement by installing plugins.

* **How To Generate Performance Graphs?**

In Nagios Core there is no inbuilt option to generate the performance graphs, We have to install pnp4nagios and add hosts and services URL’s in defination files.

* **How Do I Use Plugin X?**

 We have to download the plugins from nagios exchange https://exchange.nagios.org/. Then check the nagios plugin by running manually.

Most all plugins will display basic usage information when you execute them using ‘-h’ or ‘–help’ on the command line.

* **What Are Objects?**

Objects are all the elements that are involved in the monitoring and notification logic.

**Types of objects include:**

* 1. Services  are one of the central objects in the monitoring logic. Services are associated with hosts Attributes of a host (CPU load, disk usage, uptime, etc.)
  2. Service Groups :are groups of one or more services. Service groups can make it easier to (1) view the status of related services in the Nagios web interface and (2) simplify your configuration through the use of object tricks.
  3. Hosts  are one of the central objects in the monitoring logic.Hosts are usually physical devices on your network (servers, workstations, routers, switches, printers, etc).
  4. Host Groups   are groups of one or more hosts. Host groups can make it easier to (1) view the status of related hosts in the Nagios web interface and (2) simplify your configuration through the use of object tricks
  5. Contacts Conact information of  people involved in the notification process
  6. Contact Groups are groups of one or more contacts. Contact groups can make it easier to define all the people who get notified when certain host or service problems occur.
  7. Commands are used to tell Nagios what programs, scripts, etc. it should execute to perform ,Host and service checks and when Notifications should send etc.
  8. Time Periods are are used to control ,When hosts and services can be monitored
  9. Notification Escalations Use for escalating the the notification.
* **How To Verify Nagios Configuration?**

In order to verify your configuration, run Nagios with the -v command line option like so:

/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

If you’ve forgotten to enter some critical data or misconfigured things, Nagios will spit out a warning or error message that should point you to the location of the problem. Error messages generally print out the line in the configuration file that seems to be the source of the problem. On errors, Nagios will often exit the pre-flight check and return to the command prompt after printing only the first error that it has encountered.