**Nagios Interview Questions**

Contents

[**Nagios Interview Questions** 1](#_Toc76419799)

[1. What is Continuous Monitoring ? 3](#_Toc76419800)

[2. What is Nagios? 3](#_Toc76419801)

[3. How does Nagios work? 4](#_Toc76419802)

[4. What are Plugins in Nagios? 4](#_Toc76419803)

[5. What is NRPE (Nagios Remote Plugin Executor) in Nagios? 5](#_Toc76419804)

[6. What is meant by Nagios backend?(unable to find a relevant explanation) 5](#_Toc76419805)

[7. What do you mean by passive check in Nagios? 5](#_Toc76419806)

[8. When Does Nagios Check for external commands? 6](#_Toc76419807)

[9. What is the difference between Active and Passive check in Nagios? 6](#_Toc76419808)

[10. How does Nagios help with Distributed Monitoring? 6](#_Toc76419809)

[11. Explain Main Configuration file of Nagios and its location? 6](#_Toc76419810)

[10. Explain how Flap Detection works in Nagios? 7](#_Toc76419811)

[11. What are the three main variables that affect recursion and inheritance in Nagios? 7](#_Toc76419812)

[12. What is meant by saying Nagios is Object Oriented? 7](#_Toc76419813)

[13. What is State Stalking in Nagios? 7](#_Toc76419814)

[14. Nagios says my machine is unreachable, not down. What is the difference and how it is achieved? 9](#_Toc76419815)

[15. Explain Nagios state types? 10](#_Toc76419816)

# What is Continuous Monitoring ?

Nagios is one of the most widely used tools for Continuous Monitoring. Since organizations are now releasing software more frequently than ever so there is a dire need for a tool that can monitor the functioning of the software and provide teams with the relevant feedback. This is one of the reasons that brought Continuous Monitoring into the picture. This makes Nagios a very important tool to implement DevOps.

# What is Nagios?

Nagios is one of the monitoring tools that 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.

Few of its important features are:

|  |  |
| --- | --- |
| **Nagios Features** | |
| **Feature** | **Description** |
| ***Monitoring*** | Its powerful script APIs allow easy monitoring of in-house and custom applications, services, and systems |
| ***Visibility & Awareness*** | It provides a centralized view of the entire monitored IT infrastructure with detailed status information |
| ***Problem Remediation*** | Alert acknowledgments in Nagios, provide communication on known issues and problem response |
| ***Proactive Planning*** | Trending and capacity planning add-ons are there in Nagios to aware you about the aging infrastructure |
| ***Reporting*** | Availability reports ensure SLAs are being met & provide a record of alerts, notifications, and alert response |
| ***Customizable Code*** | Since it is an open-source software you get the full access to its source code |
| ***Large Community*** | Nagios is backed up by a community of more than 1 million+ users worldwide which provides free support |

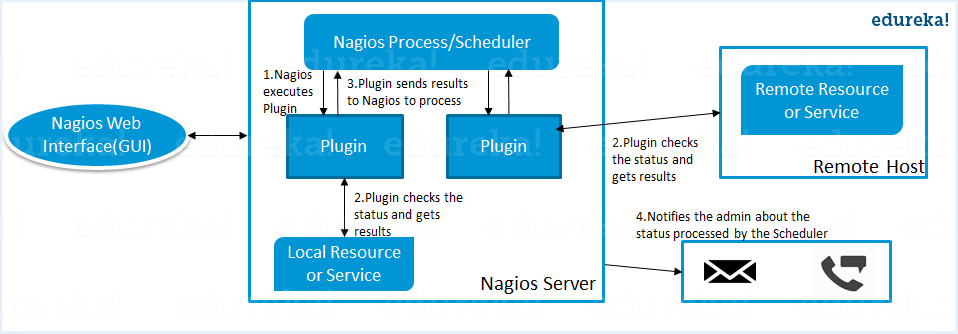
By using Nagios you can:

* 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.

# How does Nagios work?

I will advise you to follow the below explanation for this answer:

* Nagios runs on a server, usually as a daemon or service.
* Nagios periodically runs plugins residing on the same server, they contact hosts or servers on your network or on the internet.
* One can view the status information using the web interface.
* You can also receive email or SMS notifications if something happens.
* 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. Refer the diagram below:



# 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.

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.

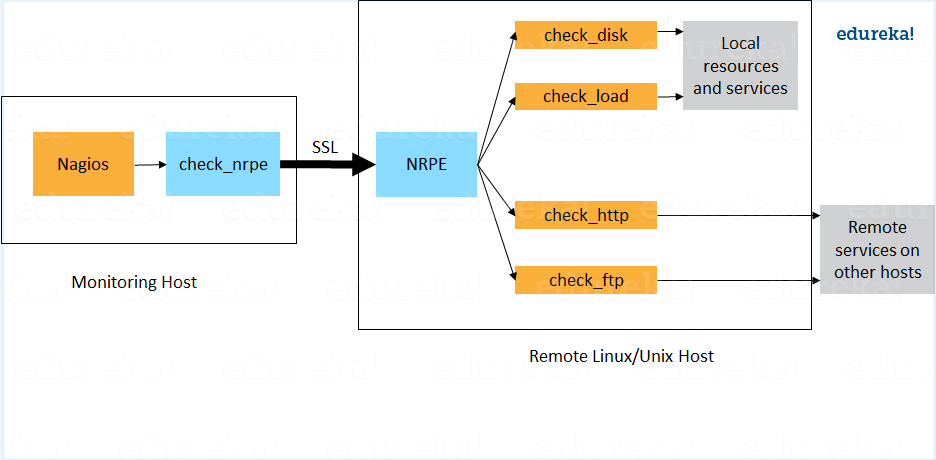
# What is NRPE (Nagios Remote Plugin Executor) in Nagios?

The NRPE 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:

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

There is a SSL (Secure Socket Layer) connection between monitoring host and the remote host as shown in the diagram.



# 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 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.

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.

# When Does Nagios Check for external commands?

Nagios check for external commands under the following conditions:

* At regular intervals specified by the command\_check\_interval option in the main configuration file or,
* 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 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:

* Asynchronous in nature and cannot be monitored effectively by polling their status on a regularly scheduled basis.
* Located behind a firewall and cannot be checked actively from the monitoring host.

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.

# 10. 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.

# 11. Explain Main Configuration file of Nagios and its location?

The main configuration file contains several 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).

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/).**

# 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.

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:

* Storing the results of the last 21 checks of the host or service analysing the historical check results and determine where state changes/transitions occur.
* Using the state transitions to determine a percent state change value (a measure of change) for the host or service.
* Comparing the percent state change value against low and high flapping thresholds
* A host or service is determined to have started flapping when its percent state change first exceeds a high flapping threshold.
* A host or service is determined to have stopped flapping when its percent state goes below a low flapping threshold.

# What are the three main variables that affect recursion and inheritance in Nagios?

The below three variables affect recursion and inheritance in Nagios:

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

# 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 is State Stalking in Nagios?

State "stalking" is a feature which is probably not going to used by most users. When enabled, it allows you to log changes in the output service and host checks even if the state of the host or service does not change. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service Check #:** | **Service State:** | **Service Check Output:** | **Logged Normally** | **Logged With Stalking** |
| x | OK | RAID array optimal | - | - |
| x+1 | OK | RAID array optimal | - | - |
| x+2 | WARNING | RAID array degraded (1 drive bad, 1 hot spare rebuilding) |  |  |
| x+3 | CRITICAL | RAID array degraded (2 drives bad, 1 host spare online, 1 hot spare rebuilding) |  |  |
| x+4 | CRITICAL | RAID array degraded (3 drives bad, 2 hot spares online) | - |  |
| x+5 | CRITICAL | RAID array failed | - |  |
| x+6 | CRITICAL | RAID array failed | - | - |
| x+7 | CRITICAL | RAID array failed | - | - |

Given this sequence of checks, you would normally only see two log entries for this catastrophe. The first one would occur at service check x+2 when the service changed from an OK state to a WARNING state. The second log entry would occur at service check x+3 when the service changed from a WARNING state to a CRITICAL state.

For whatever reason, you may like to have the complete history of this catastrophe in your log files. Perhaps to help explain to your manager how quickly the situation got out of control.

Well, if you had enabled stalking of this service for CRITICAL states, you would have events at x+4 and x+5 logged in addition to the events at x+2 and x+3.

**Why is this?**

With state stalking enabled, Nagios would have examined the output from each service check to see if it differed from the output of the previous check. If the output differed and the state of the service didn't change between the two checks, the result of the newer service check would get logged.

A similar example of stalking might be on a service that checks your web server. If the check\_http plugin first returns a WARNING state because of a 404 error and on subsequent checks returns a WARNING state because of a particular pattern not being found, you might want to know that. If you didn't enable state stalking for WARNING states of the service, only the first WARNING state event (the 404 error) would be logged and you wouldn't have any idea (looking back in the archived logs) that future WARNING states were not due to a 404, but rather some text pattern that could not be found in the returned web page.

**Should I Enable Stalking?**

First, you must decide if you have a real need to analyze archived log data to find the exact cause of a problem. You may decide you need this feature for some hosts or services, but not for all. You may also find that you only have a need to enable stalking for some host or service states, rather than all of them. For example, you may decide to enable stalking for WARNING and CRITICAL states of a service, but not for OK and UNKNOWN states.

The decision to enable state stalking for a particular host or service will also depend on the plugin that you use to check that host or service. If the plugin always returns the same text output for a particular state, there is no reason to enable stalking for that state.

**How Do I Enable Stalking?**

You can enable state stalking for hosts and services by using the stalking\_options directive in [host and service definitions](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/configobject.html).

**How Does Stalking Differ from Volatile Services?**

[Volatile services](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/volatileservices.html) are similar but will cause notifications and event handlers to run. Stalking is purely for logging purposes.

**Caveats (Warning of limitations)**

You should be aware that there are some potential pitfalls with enabling stalking. These all relate to the reporting functions found in various [CGIs](https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/3/en/cgis.html) (histogram, alert summary, etc.). Because state stalking will cause additional alert entries to be logged, the data produced by the reports will show evidence of inflated numbers of alerts.

As a general rule, I would suggest that you not enable stalking for hosts and services without thinking things through. Still, it's there if you need and want it.

# 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.

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.

A router may be defined as a parent for a server.

* Now Nagios checks for server and marks it as down.
* It then checks the parent (in our example, the router)
* If parent is also down, then server is marked as unreachable.
* If Parent is up, the server is marked as really down.

# Explain Nagios state types?

The current state of monitored services and hosts is determined by two components:

* The status of service or host i.e. OK, WARNING, UP, DOWN etc.
* The type of state the service or host is in.

There are two types of states SOFT states and HARD states.

* 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**.
* 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**.