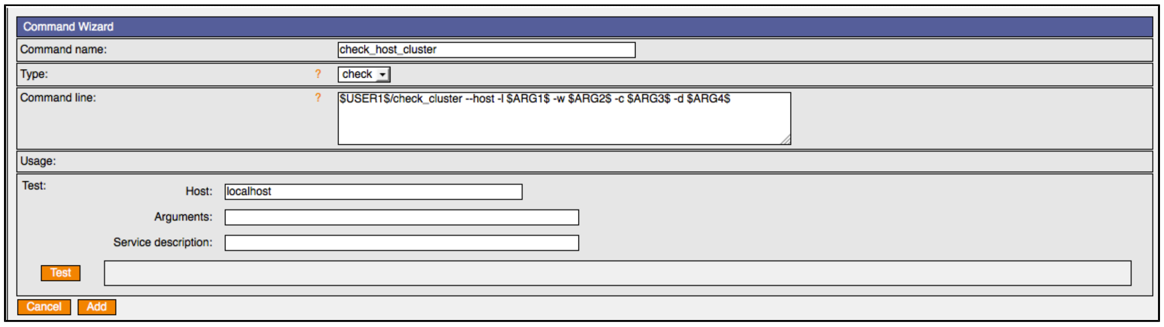
How to use check_cluster to monitor service availability of multiple services

When you are running a fault tolerant cluster, you do not necessarily want to have an alert if a small percentage of it fails. For cases like this, the check_cluster plugin works well.
The basic premise of check_cluster is that warnings and critical alerts occur only if user specified conditions for a specified number or range of host or service problems occur.
The use cases for this check include interesting sets of host-services which comprise the supporting infrastructure for an application suite. You may use such a check to inform the state of a Virtual Host representing for example your web presence. In this way you can get groupings of monitors that might otherwise require you to create HostGroups or ServiceGroups in large numbers.

The technical advantage of using the "check_cluster" plugin is that it accesses the current, in memory state of the designated Hosts and Services without having to re run those checks. The state is retrieved using Nagios Macros, a fast, efficient operation.

Before adding a check for a host or service cluster add two new custom commands to access the `check_cluster` plugin. To do this:

1. Navigate to Configuration > Command > New
    a. Select `check_cluster` from the plugin pop-up menu
    b. In the `Select resource macro:` section, select `USER1`
2. On the next screen:
    a. Enter a unique name in the `Command Name` field (suggest one is check_host_cluster and the other check_service_cluster)
    b. Append `--host` to the `Command Line` section for host checks or `--service` for service checks. This flag is necessary so that `check_cluster` properly interprets the status values passed to it. So you are making two commands.
    c. (Optional) Append a `-l` and an `$ARGn$` placeholder for supplying a label to the cluster for alerts and warnings
    d. Add a `-w` and an `$ARGn$` placeholder for setting warning thresholds
    e. Add a `-c` with an `$ARGn$` placeholder for critical thresholds
    f. Finally, add `-d` with an `$ARGn$` placeholder for the list of hosts to be checked.


Command Wizard
Command name: check_host_cluster
Type: check
Command line: $USER1$/check_cluster --host -l $ARG1$ -w $ARG2$ -c $ARG3$ -d $ARG4$
Usage:
Test:
Host: localhost
Arguments:
Service description:

To add the service name:

Monitoring Service Clusters

This is what brother Nagios produces in the configuration file from our Monarch input above:

```
define command {
    command_name          check_service_cluster
    ...
    command_line          /usr/local/groundwork/nagios/libexec/check_cluster --service -l $ARG1$ -w
$ARG2$ -c $ARG3$ -d $ARG4$
    ...
}
```

Switches and Values

-l Optional label to use when alerting. This label is only to help identify which cluster is in a troubled state.

-w Warning limits. A value set to specify when the cluster is in a warning state. It is in the form of ll:ul, where ll is the lower limit and ul is the upper limit. If it is within the range, it is considered to be not in a warning state, otherwise it is. Alternatively, you can specify it in the form @ll:ul, which inverts the condition. If ll is not specified, it is assumed to be 0. Likewise, if ul is not specified, it is taken as being equal to the maximum.

-c Critial limits. A value to set to specify when the cluster is in a critical state. The value set takes the same form as for -w.

-d A comma separated list of state values for each entry in the cluster.If a state limit exceeds the number of values supplied, the missing entries are treated as being in an unknown state.

Here is the deployed Service definition as applied to a Host, with arguments filled in:

```
define service {
    ...
    check_command        check_service_cluster!"Cluster label"
!@1!5:7!$SERVICESTATEID$:host1:servicename,$SERVICESTATEID$:host2:servicename,...
    ...
}
```

In this example, the cluster is known by the name "Cluster label", will be in a warning state if at least one service is not OK, is critical if from five to seven services (inclusive) are not OK, and has passed to it the service state identifiers for the specified "servicename" for hosts host1, host2, etc. It is important to note that $SERVICESTATEID$ is a predefined Nagios macro that will return the number corresponding to the state (0=OK, 1=Warning, 2=Critical, 3=Unknown.)

So we are telling Nagios to go get the state of a series of Host Service pairs and return a master status based on the included rules we defined under -w and -c.

Monitoring Host Clusters

Here is the command:

```
define command {
    command_name          check_host_cluster
    ...
    command_line          /usr/local/groundwork/nagios/libexec/check_cluster --host -l $ARG1$ -w
$ARG2$ -c $ARG3$ -d $ARG4$
    ...
}
```

-l Optional label to use when alerting. This label is only to help identify which cluster is in a troubled state.

-w Warning limits. A value set to specify when the cluster is in a warning state. It is in the form of ll:ul, where ll is the lower limit and ul is the upper limit. If it is within the range, it is considered to be not in a warning state, otherwise it is. Alternatively, you can specify it in the form @ll:ul, which inverts the condition. If ll is not specified, it is assumed to be 0. Likewise, if ul is not specified, it is taken as being equal to the maximum.

-c Critial limits. A value to set to specify when the cluster is in a critical state. The value set takes the same form as for -w.

-d A comma separated list of state values for each entry in the cluster.If a state limit exceeds the number of values supplied, the missing entries are treated as being in an unknown state.

Here is our service check as applied to a Host. That's a twist, a service check on the state of Hosts:

```
define service {
    ...
    check_command        check_host_cluster!"Host Cluster"
!1:3!@4!$HOSTSTATEID$:host1,$HOSTSTATEID$:host2,$HOSTSTATEID$:host3,...
    ...
}
```

In this example, the cluster is known by the name "Host Cluster", will be in a warning state if at least one and no more than three hosts are not OK, in a critical state if four or more are not OK, and is passed the state values for host1, host2, host3, etc. $HOSTSTATEID$ is a Nagios macro returning the numeric host state (0=UP, 1=DOWN, 2=UNREACHABLE.)