Ganeti customisation using hooks

Documents ganeti version 1.2

1. Introduction

In order to allow customisation of operations, ganeti will run scripts under `/etc/ganeti/hooks` based on certain rules.

This is similar to the `/etc/network/` structure present in Debian for network interface handling.

2. Organisation

For every operation, two sets of scripts are run:

- pre phase (for authorization/checking)
- post phase (for logging)

Also, for each operation, the scripts are run on one or more nodes, depending on the operation type.

Note that, even though we call them scripts, we are actually talking about any executable.

2.1. *pre* scripts

The *pre* scripts have a definite target: to check that the operation is allowed given the site-specific constraints. You could have, for example, a rule that says every new instance is required to exists in a database; to implement this, you could write a script that checks the new instance parameters against your database.

The objective of these scripts should be their return code (zero or non-zero for success and failure). However, if they modify the environment in any way, they should be idempotent, as failed executions could be restarted and thus the script(s) run again with exactly the same parameters.

Note that if a node is unreachable at the time a hooks is run, this will not be interpreted as a deny for the execution. In other words, only an actual error returned from a script will cause abort, and not an unreachable node.

Therefore, if you want to guarantee that a hook script is run and denies an action, it's best to put it on the master node.

2.2. *post* scripts

These scripts should do whatever you need as a reaction to the completion of an operation. Their return code is not checked (but logged), and they should not depend on the fact that the *pre* scripts have been run.

2.3. Naming

The allowed names for the scripts consist of (similar to run-parts(8)) upper and lower case, digits, underscores and hyphens. In other words, the regexp `^[a-zA-Z0-9_-]+$`. Also, non-executable scripts will be ignored.

2.4. Order of execution

On a single node, the scripts in a directory are run in lexicographic order (more exactly, the python string comparison order). It is advisable to implement the usual *NN-name* convention where *NN* is a two digit number.

For an operation whose hooks are run on multiple nodes, there is no specific ordering of nodes with regard to hooks execution; you should assume that the scripts are run in parallel on the target nodes (keeping on each node the above specified ordering). If you need any kind of inter-node synchronisation, you have to implement it yourself in the scripts.

2.5. Execution environment

The scripts will be run as follows:

- no command line arguments
- no controlling tty
- `stdin` is actually `/dev/null`
- `stdout` and `stderr` are directed to files
- the `PATH` is reset to `/sbin:/bin:/usr/sbin:/usr/bin`
- the environment is cleared, and only ganeti-specific variables will be left

All informations about the cluster is passed using environment variables. Different operations will have sligthly different environments, but most of the variables are common.

2.6. Operation list

Table 1. Operation list

Operation ID	Directory prefix	Description	Command	Supported env. variables	*pre* hooks	*post* hooks
OP_INIT_CLUSTER	cluster-init	Initialises the cluster	gnt-cluster init	CLUSTER, MASTER	master node, cluster name	
OP_MASTER_FAILOVER	master-failover	Changes the master	gnt-cluster master-failover	OLD_MASTER, NEW_MASTER	the new master	all nodes
OP_ADD_NODE	node-add	Adds a new node to the cluster	gnt-node add	NODE_NAME, NODE_PIP, NODE_SIP	all existing nodes	all existing nodes plus the new node
OP_REMOVE_NODE	node-remove	Removes a node from the cluster	gnt-node remove	NODE_NAME	all existing nodes except the removed node	
OP_INSTANCE_ADD	instance-add	Creates a new instance	gnt-instance add	INSTANCE_NAME, INSTANCE_PRIMARY, INSTANCE_SECONDARIES, DISK_TEMPLATE, MEM_SIZE, DISK_SIZE, SWAP_SIZE, VCPUS, INSTANCE_IP, INSTANCE_ADD_MODE, SRC_NODE, SRC_PATH, SRC_IMAGE	master node, primary and secondary nodes	
OP_BACKUP_EXPORT	instance-export	Export the instance	gnt-backup export	INSTANCE_NAME, EXPORT_NODE, EXPORT_DO_SHUTDOWN		
OP_INSTANCE_START	instance-start	Starts an instance	gnt-instance start	INSTANCE_NAME, INSTANCE_PRIMARY, INSTANCE_SECONDARIES, FORCE		

Operation ID	Directory prefix	Description	Command	Supported env. variables	*pre* hooks	*post* hooks
OP_INSTANCE_SHUTDOWN	instance-shutdown	Stops an instance	`gnt-instance shutdown`	`INSTANCE_NAME`, `INSTANCE_PRIMARY`, `INSTANCE_SECONDARIES`		
OP_INSTANCE_MODIFY	instance-modify	Modifies the instance parameters.	`gnt-instance modify`	`INSTANCE_NAME`, `MEM_SIZE`, `VCPUS`, `INSTANCE_IP`		
OP_INSTANCE_FAILOVER	instance-failover	Failover an instance	`gnt-instance start`	`INSTANCE_NAME`, `INSTANCE_PRIMARY`, `INSTANCE_SECONDARIES`, `IGNORE_CONSISTENCY`		
OP_INSTANCE_REMOVE	instance-remove	Remove an instance	`gnt-instance remove`	`INSTANCE_NAME`, `INSTANCE_PRIMARY`, `INSTANCE_SECONDARIES`	master node	
OP_INSTANCE_ADD_MDDRBD	mirror-add	Adds a mirror component	`gnt-instance add-mirror`	`INSTANCE_NAME`, `NEW_SECONDARY`, `DISK_NAME`		
OP_INSTANCE_REMOVE_MDDRBD	mirror-remove	Removes a mirror component	`gnt-instance remove-mirror`	`INSTANCE_NAME`, `OLD_SECONDARY`, `DISK_NAME`, `DISK_ID`		
OP_INSTANCE_REPLACE_DISKS	mirror-replace	Replace all mirror components	`gnt-instance replace-disks`	`INSTANCE_NAME`, `OLD_SECONDARY`, `NEW_SECONDARY`		

2.7. Environment variables

Note that all variables listed here are actually prefixed with `GANETI_` in order to provide a different namespace.

2.7.1. Common variables

This is the list of environment variables supported by all operations:

HOOKS_VERSION

Documents the hooks interface version. In case this doesnt match what the script expects, it should not run. The documents conforms to the version `1`.

HOOKS_PHASE

one of `PRE` or `POST` denoting which phase are we in.

CLUSTER

the cluster name

MASTER

the master node

OP_ID

one of the `OP_*` values from the table of operations

OBJECT_TYPE

one of `INSTANCE`, `NODE`, `CLUSTER`, showing the target of the operation.

2.7.2. Specialised variables

This is the list of variables which are specific to one or more operations.

INSTANCE_NAME

The name of the instance which is the target of the operation.

INSTANCE_DISK_TYPE

The disk type for the instance.

INSTANCE_DISK_SIZE

The (OS) disk size for the instance.

INSTANCE_OS

The name of the instance OS.

INSTANCE_PRIMARY

The name of the node which is the primary for the instance.

INSTANCE_SECONDARIES

Space-separated list of secondary nodes for the instance.

NODE_NAME

The target node of this operation (not the node on which the hook runs).

NODE_PIP

The primary IP of the target node (the one over which inter-node communication is done).

NODE_SIP

The secondary IP of the target node (the one over which drbd replication is done). This can be equal to the primary ip, in case the cluster is not dual-homed.

OLD_MASTER
NEW_MASTER

The old, respectively the new master for the master failover operation.

FORCE

This is provided by some operations when the user gave this flag.

IGNORE_CONSISTENCY

The user has specified this flag. It is used when failing over instances in case the primary node is down.

MEM_SIZE, DISK_SIZE, SWAP_SIZE, VCPUS

The memory, disk, swap size and the number of processor selected for the instance (in **gnt-instance add** or **gnt-instance modify**).

INSTANCE_IP

If defined, the instance IP in the **gnt-instance add** and **gnt-instance set** commands. If not defined, it means that no IP has been defined.

DISK_TEMPLATE

The disk template type when creating the instance.

INSTANCE_ADD_MODE

The mode of the create: either `create` for create from scratch or `import` for restoring from an exported image.

SRC_NODE, SRC_PATH, SRC_IMAGE

In case the instance has been added by import, these variables are defined and point to the source node, source path (the directory containing the image and the config file) and the source disk image file.

DISK_NAME

The disk name (either `sda` or `sdb`) in mirror operations (add/remove mirror).

DISK_ID

The disk id for mirror remove operations. You can look this up using **gnt-instance info**.

NEW_SECONDARY

The name of the node on which the new mirror componet is being added. This can be the name of the current secondary, if the new mirror is on the same secondary.

OLD_SECONDARY

The name of the old secondary. This is used in both **replace-disks** and **remove-mirror**. Note that this can be equal to the new secondary (only **replace-disks** has both variables) if the secondary node hasn't actually changed).

EXPORT_NODE

The node on which the exported image of the instance was done.

EXPORT_DO_SHUTDOWN

This variable tells if the instance has been shutdown or not while doing the export. In the "was shutdown" case, it's likely that the filesystem is consistent, whereas in the "did not shutdown" case, the filesystem would need a check (journal replay or full fsck) in order to guarantee consistency.