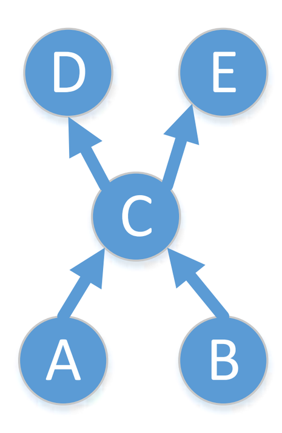
The following diagrams depict the flow of control and changes to database tables while serving a Heat stack request. This is part of persist-graph-and-resource-versioning specification: https://review.openstack.org/#/c/123749/

Stack create

For our understanding let's create a stack that has following graph.



resource	needed_by	stack_id	traversed
A	C	1	F
B	C	1	F
C	D	1	F
C	E	1	F
D	-	1	F
E	-	1	F

Dependency graph table

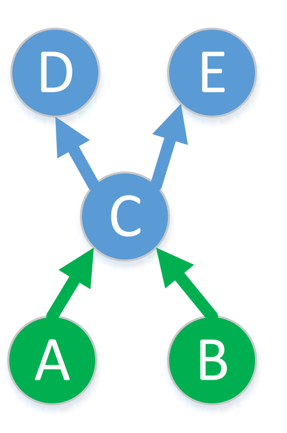
name	physical_id	rsrc_defn	version	action	status	sha1

Resource Table

id	name	action	status	raw_template_id
1	S	CREATE	IN_PROGRESS	1

Stack Table

1. Upon receiving a stack request, the template is parsed, validated and graph edges are stored in DB Edges are marked as not traversed.
2. Engine issues a DB API request to retrieve next set of ready nodes for stack. Stack ID 1 in this example.
3. Ready nodes (resources that need to be converged) is computed in following manner:
   column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
   (A, B, C, D, E) – (C, D, E) = (A, B)
   Returns (A, B)
4. Engine triggers converge for A and B



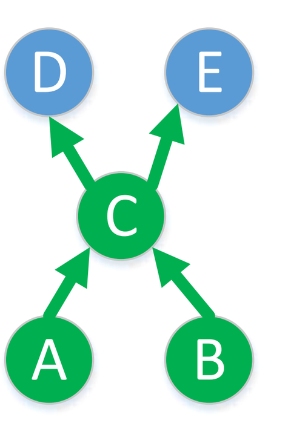
resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	D	1	F
C	E	1	F
D	-	1	F
E	-	1	F

Dependency Graph table

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t

Resource table

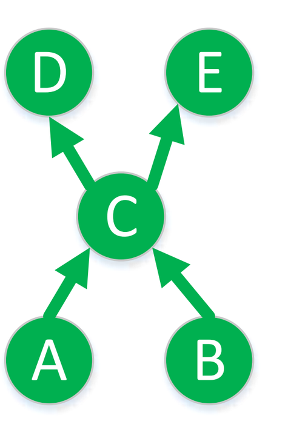
After A and B are created, engine calls "Get next ready nodes for stack id 1"
column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
i.e. (C, D, E) – (D, E) = (C)
Returns (C); Engine triggers converge for C



resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	D	1	T
C	E	1	T
D	-	1	F
E	-	1	F

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C	300	{defn-C-0}	0	CREATE	COMPLETE	i76y

After C is created, engine calls "Get next ready nodes for stack id 1"
column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
i.e. (D, E) – () = (D, E)
Returns (D, E); Engine triggers converge for D and E



resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	D	1	T
C	E	1	T
D	-	1	T
E	-	1	T

Dependency Table

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C	300	{defn-C-0}	0	CREATE	COMPLETE	i76y
D	400	{defn-D-0}	0	CREATE	COMPLETE	d67y
E	500	{defn-E-0}	0	CREATE	COMPLETE	d9p0

Resource Table

id	name	action	status	raw_template_id
1	S	CREATE	COMPLETE	1

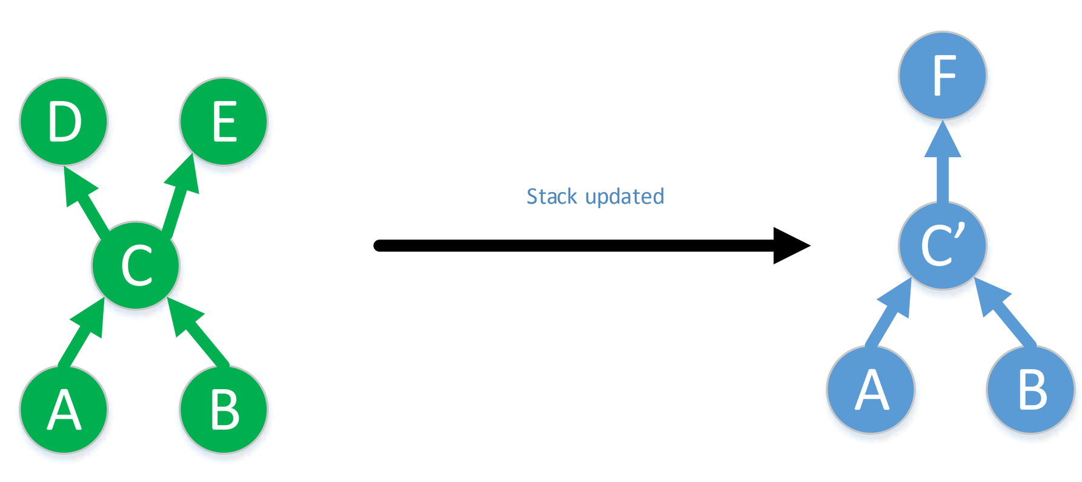
Stack Table

id	template	env	files	predecessor
1	{template-s}	{user-env-s}	{"f":"file:///f}	NULL

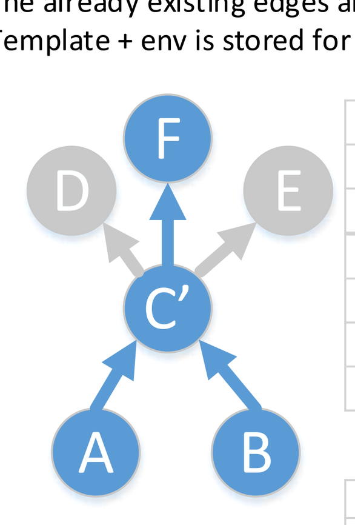
Raw_template table

After D and E are created, engine gets next set of ready nodes.
Since there are no more edges to be traversed, the stack is marked as CREATE COMPLETE.

Stack update



Stack is updated:
Resources D and E are removed, C is updated and F is added. C' indicates C has been updated.
The graph is updated in the database to include the new edges and overwrite the existing edges. The new edges are added to the graph, i.e. edges (A, C), (B, C), (C, F), (F, -). Note that in previous graph, C has two edges, but now there is only one edge, so the two entries are replace with one: (C, D) and (C, E) is replaced with (C, F). Edges (D, -) and (E, -) are not found in new graph, so they are not updated and they remain. The already existing edges are not touched and the entire graph is marked as not traversed.
Template + env is stored for this update and old is made predecessor of new template.



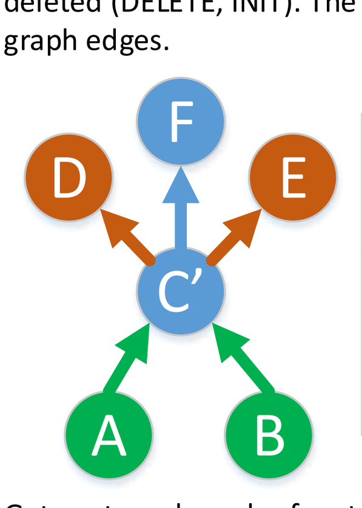
resource	needed_by	stack_id	traversed
A	C	1	F
B	C	1	F
C	F	1	F
D	-	1	F
E	-	1	F
F	-	1	F

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C	300	{defn-C-0}	0	CREATE	COMPLETE	i76y
D	400	{defn-D-0}	0	CREATE	COMPLETE	d67y
E	500	{defn-E-0}	0	CREATE	COMPLETE	d9p0

id	name	action	status	raw_template_id
1	S	UPDATE	IN_PROGRESS	2

id	template	env	files	predecessor
1	{template-s}	{user-env-s}	{"f":"file:///f}	NULL
2	{template-2-s}	{user-env-2-s}	{"f":"file:///f1}	1

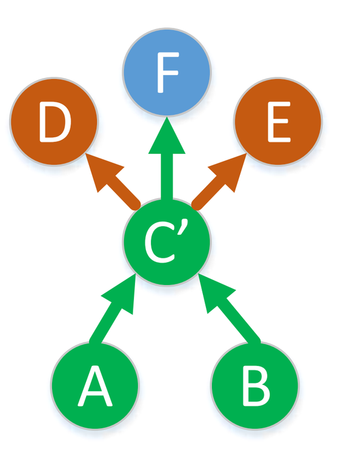
Get next ready nodes for stack id 1
column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
i.e. (A, B, C, D, E, F) – (C, F) = (A, B, D, E)
Returns (A, B, D, E); Engine triggers converge for A and B. Engine will compute SHA1 hash of current definition of A and B from current template + environment and compares with SHA1 of version 0.
Since A and B has not changed, nothing is done at resource level. D and E are not found in the new template, so a new version is created and marked as deleted (DELETE, INIT). The physical resource IDs are copied to delete versions to delete the physical resources before deleting the all the versions and graph edges.



resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	F	1	F
D	-	1	T
E	-	1	T
F	-	1	F

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C	300	{defn-C-0}	0	CREATE	COMPLETE	i76y
D		{defn-D-0}	0	CREATE	COMPLETE	d67y
E		{defn-E-0}	0	CREATE	COMPLETE	d9p0
D	400	{defn-D-0}	1	DELETE	INIT	d67y
E	500	{defn-E-0}	1	DELETE	INIT	d9p0

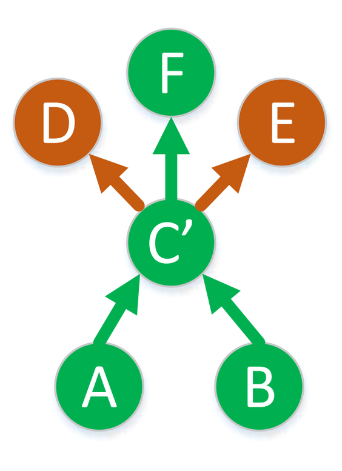
Get next ready nodes for stack id 1
column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
i.e. (C, F) – (D, E, F) = (C)
Returns (C); Engine triggers converge for C. Since C has changed. SHA1 hash will not match. Engine will then create next version i.e. version 1 for C and updates it. If update-in-place is done, then the physical resource ID is moved from old version to new version.
Edges originating out of C is marked as traversed.



resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	F	1	T
D	-	1	T
E	-	1	T
F	-	1	F

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C		{defn-C-0}	0	CREATE	COMPLETE	i76y
D		{defn-D-0}	0	CREATE	COMPLETE	d67y
E		{defn-E-0}	0	CREATE	COMPLETE	d9p0
D	400	{defn-D-0}	1	DELETE	INIT	d67y
E	500	{defn-E-0}	1	DELETE	INIT	d9p0
C	300	{defn-C-1}	1	UPDATE	COMPLETE	n8o9

Get next ready nodes for stack id 1
column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
i.e. (F) – () = (F)
Returns (F); for F there is no old version in DB, so a new resource is created for F..



resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	F	1	T
D	-	1	T
E	-	1	T
F	-	1	T

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C		{defn-C-0}	0	CREATE	COMPLETE	i76y
D		{defn-D-0}	0	CREATE	COMPLETE	d67y
E		{defn-E-0}	0	CREATE	COMPLETE	d9p0
D	400	{defn-D-0}	1	DELETE	INIT	d67y
E	500	{defn-E-0}	1	DELETE	INIT	d9p0
C	300	{defn-C-1}	1	UPDATE	COMPLETE	n8o9
F	600	{defn-F-0}	0	CREATE	COMPLETE	f78u

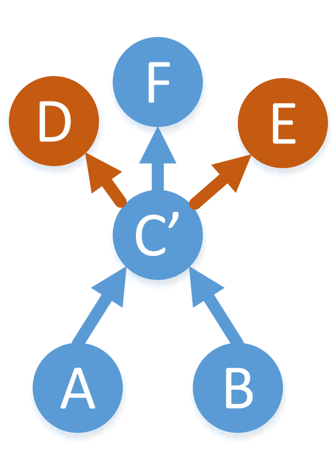
Get next ready nodes for stack id 1
column(resource) - column(needed_by) from graph table where stack_id = 1 and traversed = False
Returns (); No resource to be updated.

Now, the engine will traverse the graph in reverse order and resources marked as deleted are deleted from physical world and older versions of resources are also deleted. The stack status is set to GC_IN_PROGRESS and GC is triggered.

Stack delete old resources

Stack status is set to GC_IN_PROGRESS. The traversal flag is set to false.

GC_IN_PROGRESS indicates that graph has to be traversed in reverse order.

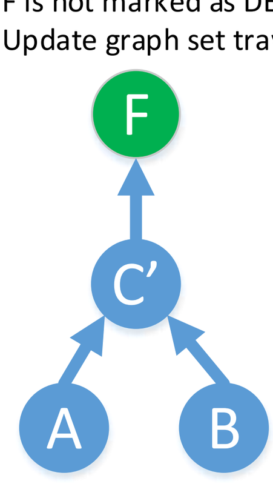


resource	needed_by	stack_id	traversed
A	C	1	F
B	C	1	F
C	F	1	F
D	-	1	F
E	-	1	F
F	-	1	F

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C		{defn-C-0}	0	CREATE	COMPLETE	i76y
D		{defn-D-0}	0	CREATE	COMPLETE	d67y
E		{defn-E-0}	0	CREATE	COMPLETE	d9p0
D	400	{defn-D-0}	1	DELETE	INIT	d67y
E	500	{defn-E-0}	1	DELETE	INIT	d9p0
C	300	{defn-C-1}	1	UPDATE	COMPLETE	n8o9
F	600	{defn-F-0}	0	CREATE	COMPLETE	f78u

id	name	action	status	raw_template_id
1	S	UPDATE	GC_IN_PROGRESS	2

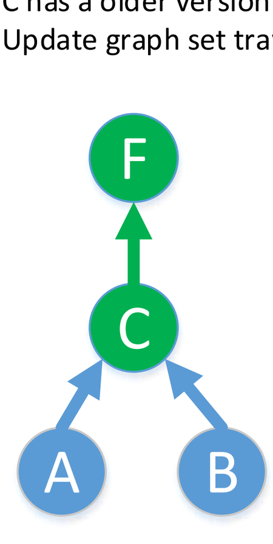
Get next ready nodes for stack id 1
Since status is GC_IN_PROGRESS, graph is traversed in reverse order.
column(resource) from graph table where stack_id = 1 and traversed = False and needed_by in (resource where stack_id = 1 and traversed = True OR Null)
i.e. resources where needed_by is null = (D, E, F)
D and E are marked for deletion; all their versions with physical resources are deleted (convergence jobs for resources with physical ID) and edges are also deleted. Edges are deleted for all resources marked as (DELETE, INIT). For D and E, all the edges originating from and merging into them are deleted.
F is not marked as DELETE and has no older versions, nothing is done. All the edges originating from F are marked as traversed.
Update graph set traversed = True where resource is F.



resource	needed_by	stack_id	traversed
A	C	1	F
B	C	1	F
C	F	1	F
F	-	1	T

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C		{defn-C-0}	0	CREATE	COMPLETE	i76y
C	300	{defn-C-1}	1	UPDATE	COMPLETE	n8o9
F	600	{defn-F-1}	0	CREATE	COMPLETE	f78u

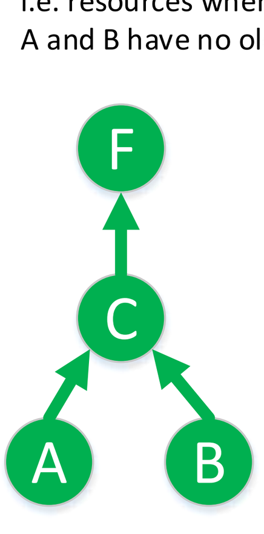
Get next ready nodes for stack id 1
Since status is GC_IN_PROGRESS, graph is traversed in reverse order.
column(resource) from graph table where stack_id = 1 and traversed = False and needed_by in (resource where stack_id = 1 and traversed = True OR Null)
i.e. resources where needed_by resources are traversed = (C)
C has a older version which needs to be deleted. It is deleted, and as it doesn't have a physical_id nothing is done in physical world.
Update graph set traversed = True where resource is C.



resource	needed_by	stack_id	traversed
A	C	1	F
B	C	1	F
C	F	1	T
F	-	1	T

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C	300	{defn-C-1}	1	UPDATE	COMPLETE	n8o9
F	600	{defn-F-1}	0	CREATE	COMPLETE	f78u

Get next ready nodes for stack id 1
Since status is GC_IN_PROGRESS, graph is traversed in reverse order.
column(resource) from graph table where stack_id = 1 and traversed = False and needed_by in (resource where stack_id = 1 and traversed = True OR Null)
i.e. resources where needed_by resources are traversed = (A, B)
A and B have no older versions and they are not marked as deleted, so nothing is done. The graph is updated as traversed.



resource	needed_by	stack_id	traversed
A	C	1	T
B	C	1	T
C	F	1	T
F	-	1	T

name	physical_id	rsrc_defn	version	action	status	sha1
A	100	{defn-A-0}	0	CREATE	COMPLETE	1a6k
B	200	{defn-B-0}	0	CREATE	COMPLETE	8b6t
C	300	{defn-C-1}	1	UPDATE	COMPLETE	n8o9
F	600	{defn-F-1}	0	CREATE	COMPLETE	f78u

Since the graph is completely traversed, it's status is marked as COMPLETE and old template is removed from the template table and new template's predecessor is marked as NULL.

id	name	action	status	raw_template_id
1	S	UPDATE	COMPLETE	2

id	template	env	files	predecessor
2	{template-2-s}	[user-env-2-s]	{"f":"file:///f1}	NULL