A universal profile deployer tool

author: noelmcloughlin
Revision: v.01
Date: November 2018

Table of Contents

Problem Description

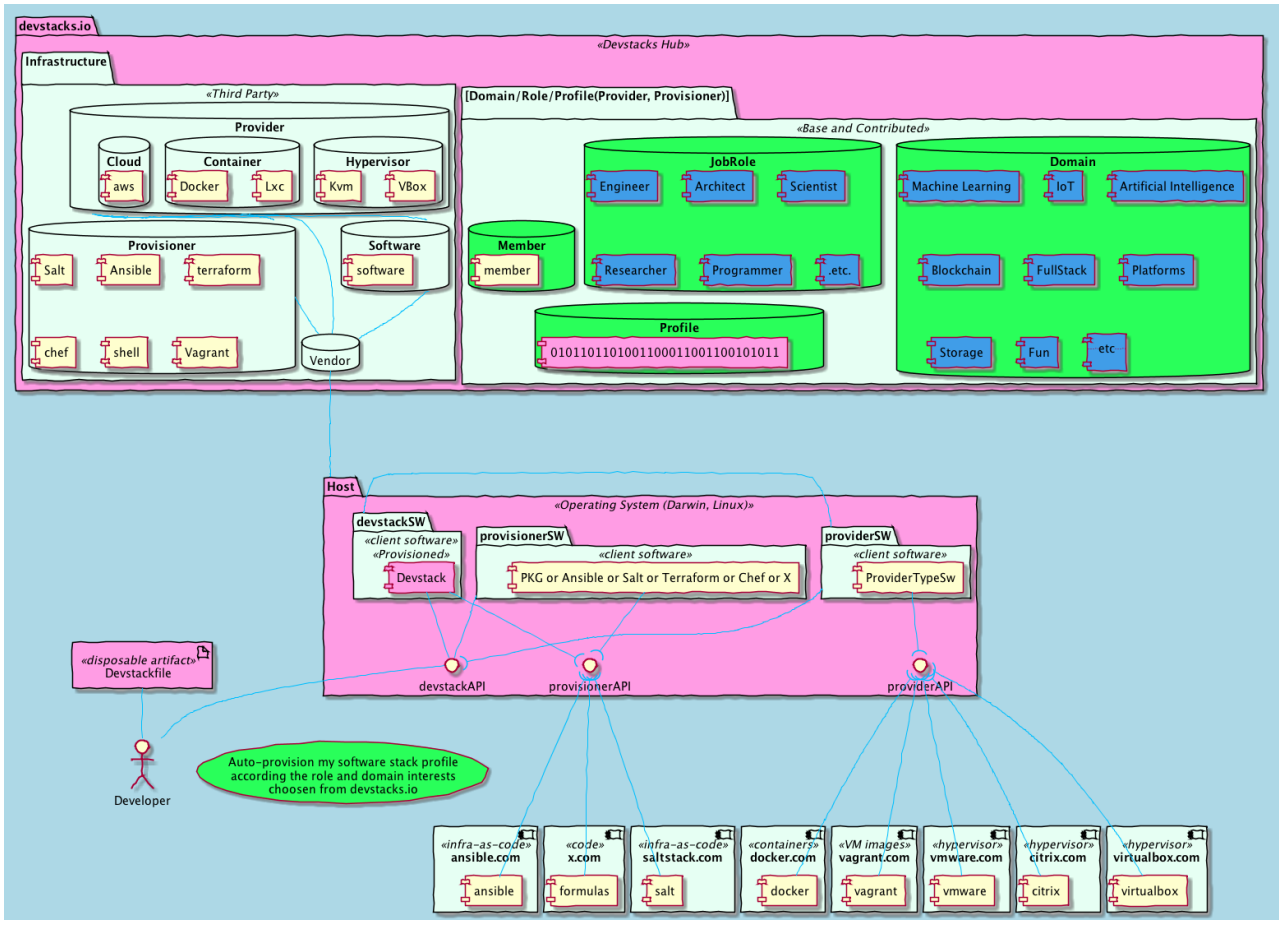
TL;DR: user automation is about domains and associated profiles.

Technology for machine (hypervisor) and operating-system (containerization) virtualization are useful – S/W and execution environment bundled together, and the packages can be shared at online "hubs". Providers such as vagrant and docker support VM packages and containers respectively. Provisioners such as shell scripts, and infrastructure-as-code solutions deploy these packages.

Imagine a *universal automation tool*. Users can visit an "devstacks.io" portal to view available software stack categories, and ***browse***, **add, delete,** or **select** a domain; select presents jobroles. User can *browse*, add, or delete a jobrole; select presents available profiles; user can **upload** or select a profile; select retrieves the command to deploy a Developer software stack composed of-

- Profile (wanted software and configuration)
- Provisioner (technology)
- Provider (technology)

Figure 1: Architectural view



Entity Relationship Model

Entity Relationship (ER) modeling is the preferred methodology to model outline data structure for logical view of the entire database independent of physical RDBMS used.

Entities

The architectural diagram suggests these entities:

- ***Vendor***
- ***Provider***
- **Provisioner**
- **Software**
- **Domain**
- **JobRole**
- **Profile**
- **tbl_member**

The *Developer* is an "actor", interacting with system, so not an entity.

Entity Relationships

Identify entity relationships, and multiplicity constraints (cardinality).

- Vendor 1..* vends 0..1 Software
  - Each Vendor provides zero or many Software.
  - Each Software is vended by one or many Vendor.

- Vendor 1..* vends 0..* Provider
  - Each Vendor vends zero or many Provider.
  - Each Provider is vended by one or many Vendor.

- Vendor 1..* vends 0..1 Provisioner
  - Each Vendor provides zero or many Provisioner.
  - Each Provisioner is vended by one or many Vendor.

- Domain 1..* holds 1..* JobRole.
  - Each Domain holds one or many JobRole.
  - Each JobRole is held by one or many Domain.

- Domain 1..* holds 1..* Profile
  - Each JobRole holds one or many Profile.
  - Each Profile is held by one or many JobRole.

- Profile 0..* uses 1..* Software
    - Each Profile uses one or many Software.
    - Each Software is used by zero or many Profile.

- Profile 0..* uses 1..* Provisioner
    - Each Profile uses one or many Provisioner.
    - Each Provisioner is used by zero or many Profile.

- Profile 0..* uses 1..* Provider
    - Each Profile uses one or many Provider.
    - Each Provider is used by zero or many Profile.

- Member 0..* likes 1..* Vendor
    - Each tbl_member likes one or many Vendor.
    - Each Vendor is liked by zero or many tbl_member.

- Member 0..* likes 1..* Software.
    - Each tbl_member likes one or many Software.
    - Each Software is liked by zero or many tbl_member.

- Member 0..* likes 1..* Profile
    - Each tbl_member likes one or many Profile.
    - Each Profile is liked by zero or many tbl_member.

- Member 0..* likes 1..* JobRole
    - Each tbl_member likes one or many JobRole.
    - Each JobRole is liked by zero or many tbl_member.

- Member 0..* likes 1..* Domain.
    - Each tbl_member likes one or many Domain.
    - Each Domain is liked by zero or many tbl_member.

- Member 0..* likes 1..* Provider.
    - Each tbl_member likes one or many Provider.
    - Each Provider is liked by zero or many tbl_member.

- Member 0..* likes 1..* Provisioner.
    - Each tbl_member likes one or many Provisioner.
    - Each Provisioner is liked by zero or many tbl_member.

- Provisioner 0..* supports 1..* Provider.
    - Each Provisioner supports one or more Provider.
    - Each Provider is supported by one or more Provisioner.

Entity and Relationship attributes

Now associate attributes with entity types and relationships.

Entity Type attributes

Each Entity type has the following attributes.

- **Vendors:** vendor_id, Name, ApiVersion, vendor_role
- **Software**: software_id, Name, Type, ApiVersion, software_roles, Vendors, Profiles, members
- **Profile**: profile_id, Name, Type, Url, ApiVersion, Providers, Software, Provisioners, members
- **JobRole**: jobrole_id, Name, Type, ApiVersion, Profiles, Domains, members
- **Domain**: domain_id, Name, Type, ApiVersion, JobRoles, members
- **Provisioner**: ProvisionerID, Name, Type, ApiVersion, Providers, members
- **Provider**: provider_id, Name, Type, ApiVersion, Provisioners, Profiles, members
    - **CloudProvider**: cloudId
    - **ContainerProvider**: ContainerId
    - **HypervisorProvider**: HypervisorId
    - **OsProvider**: software_id
- **Member**: member_id, Name (fName, lName), ApiVersion, member_roles, Profiles, JobRoles, Domains, Softwares, Providers, Provisioners

Relationship Type Attributes

Some Entity relationships will be recorded. tbl_members may have systems roles (“student”, “recruiter”, “expert”, “creator”, “user”, “manager”, etc). The same rationale applies to Software and Vendors who can have system roles defined by Domain experts.

- **tbl_member_role:** fk_member_role_id, Name, ApiVersion
- **tbl_software_role:** SwRoleId, Name, ApiVersion
- **tbl_vendor_role:** tbl_vendor_roleId, Name, ApiVersion

Attribute Domains

Record allowable set of values, size, and format for attributes in the Data Dictionary.

Table	Field	Format	Size	Set of Values
tbl_member_role				
	member_role_id	char	8	1..n
	Name	char	64	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n
tbl_software_role				
	SwRoleId	char	8	1..n
	Name	char	64	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n
tbl_vendor_role				
	VenderRoleId	char	8	1..n
	Name	char	64	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n
tbl_member				
	member_id	Char	8	1..n
	fName	Char	24	a-z,A-Z,0-n , punc
	lName	Char	24	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n, .
	Profiles, JobRoles, Domains, Software, Providers, Provisioners			Multivalued

Software				
	software_id	char	8	1..n
	Name	char	64	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n
	Vendors, Profiles, tbl_members			Multivalued
tbl_vendor				
	vendor_id	Char	8	1..n
	Name	Char	48	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n
tbl_profile				
	profile_id	char	8	1..n
	Name	char	128	a-z,A-Z,0-n , punc
	ApiVersion	char	16	1..n, .
	Url	char	256	a-z,A-Z,0-n , punc
	Providers, Software, Provisioners, tbl_members			Multivalued
tbl_jobrole				
	jobrole_id	char	8	1..n
	Name	char	48	a-z,A-Z,0-n , punc
	Type	char	8	a-z,Z-Z,0-n, Null
	ApiVersion	char	16	1..n, .
	Profiles, Domains, tbl_members			Multivalued
tbl_domain				
	domain_id	Char	8	1..n
	Name	char	48	a-z,A-Z,0-n , punc
	Type	char	8	a-z,Z-Z,0-n, Null
	ApiVersion	char	16	1..n, .
	JobRoles, tbl_members			Multivalued
tbl_provisioner				
	provisioner_id	Char	8	1..n
	Name	Char	48	a-z,A-Z,0-n , punc
	Type	Char	8	a-z,Z-Z,0-n, Null
	ApiVersion	char	16	1..n, .
	Providers, tbl_members			Multivalued
tbl_provider				
	provider_id	Char	8	1..n
	Name	Char	48	a-z,A-Z,0-n , punc
	Type	Char	8	a-z,Z-Z,0-n, Null
	ApiVersion	char	16	1..n, .
	Provisioners, Profiles, tbl_members, CloudId, HypervisorID, ContainerId, OsID			Multivalued

Entity Key Attributes

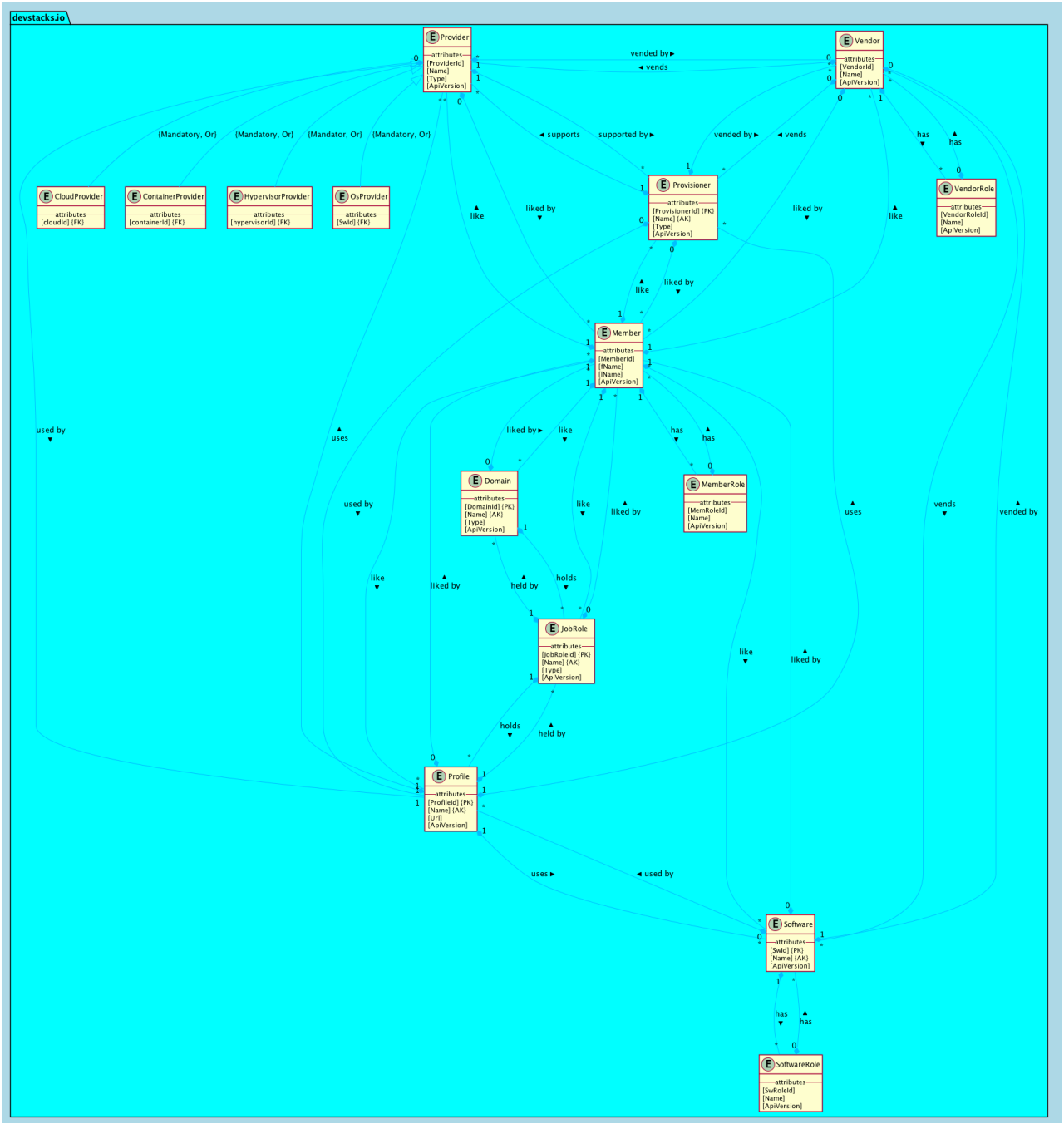
Record the candidate, primary, and alternate key attributes for all entity types.

- **tbl_member_role** (fk_member_role_id, Name, ApiVersion)
  - Candidate keys: fk_member_role_id, Name
  - Primary key: fk_member_role_id
  - Alternate keys: Name
- **tbl_software_role** (SwRoleId, Name, ApiVersion)
  - Candidate keys: SwRoleId, Name
  - Primary key: SwRoleId
  - Alternate keys: Name
- **tbl_vendor_role** (tbl_vendor_roleId, Name, ApiVersion)
  - Candidate keys: tbl_vendor_roleId, Name
  - Primary key: tbl_vendor_roleId
  - Alternate keys: Name
- **tbl_member** (member_id, fName, lName, ApiVersion)
  - Candidate keys: member_id, Name
  - Primary key: member_id
  - Alternate key: Name
- **tbl_vendor:** (vendor_id, Name, ApiVersion)
  - Candidate keys: vendor_id, Name

    - Primary key: vendor_id
    - Alternate key: Name
- **tbl_software** (software_id, Name, ApiVersion)
    - Candidate keys: software_id, Name
    - Primary key: software_id
    - Alternate key: Name
- **tbl_profile** (profile_id, Name, Url, ApiVersion)
    - Candidate keys: profile_id, Name
    - Primary key: profile_id
    - Alternate key: Name
- **tbl_jobrole** (jobrole_id, Name, Type, ApiVersion)
    - Candidate keys: jobrole_id, Name
    - Primary key: jobrole_id
    - Alternate key: Name
- **tbl_domain** (domain_id, Name, Type, ApiVersion)
    - Candidate keys: domain_id, Name
    - Primary key: domain_id
    - Alternate key: Name
- **tbl_provisioner** (ProvisionerID, Name, Type, ApiVersion)
    - Candidate keys: provisioner_id, Name
    - Primary key: provisioner_id
    - Alternate key: Name
- **tbl_provider** (provider_id, Name, Type, ApiVersion)
    - Candidate keys: provider_id, Name
    - Primary key: provider_id
    - Alternate key: Name

Enhanced ER Diagram

The ER Diagram displays the conceptual entity types, relations, cardinality of associations, and superclass/subclass associations documented.



Entity Relationships and Referential Integrity Constraints

Record the set of relations (multivalued attribute qualification) for all entity types.

Vendors **supply** one or many Software, Provisioner, and Provider. These mandatory one-to-many relationships require new entity relations.

- **tbl_vends_software**(vendor_id, software_id)
  - Primary key vendor_id, software_id
  - Foreign key fk_vendor_id references tbl_vendor(vendor_id)

        - On update cascade, on delete no action
    - Foreign key fk_software_id references tbl_software(software_id)
        - On update cascade, on delete no action
- **tbl_vends_provider** (vendor_id, provider_id)
    - Primary key vendor_id, provider_id
    - Foreign key fk_vendor_id references tbl_vendor(vendor_id)
        - On update cascade, on delete no action
    - Foreign key fk_provider_id references tbl_provider(provider_id)
        - On update cascade, on delete no action
- **tbl_vends_provisioner** (vendor_id, provisioner_id)
    - Primary key vendor_id, provisioner_id
    - Foreign key fk_vendor_id references tbl_vendor(vendor_id)
        - On update cascade, on delete no action
    - Foreign key fk_provisioner_id references tbl_provisioner(provisioner_id)
        - On update cascade, on delete no action

Domains **hold** one or many JobRole. This mandatory one-to-many relationship requires a new entity relation.

- **tbl_holds_jobrole** (domain_id, jobrole_id)
    - Primary key jobrole_id, domain_id
    - Foreign key fk_jobrole_id references tbl_jobrole(jobrole_id)
        - On update cascade, on delete no action
    - Foreign key fk_domain_id references Domain (domain_id)
        - On update cascade, on delete no action

JobRoles **hold** one or many Profile. This mandatory one-to-many relationship needs a new entity relation.

- **tbl_holds_profile** (jobrole_id, profile_id)
    - Primary key jobrole_id, profile_id
    - Foreign key fk_jobrole_id references tbl_jobrole(jobrole_id)
        - On update cascade, on delete no action
    - Foreign key fk_profile_id references tbl_profile(profile_id)
        - On update cascade, on delete no action

Profiles **consume** one or many Software, Provisioner, and Providers. These mandatory one-to-many relationships needs new entity relations.

- **tbl_uses_software** (profile_id, software_id)
    - Primary key profile_id, software_id
    - Foreign key fk_profile_id references tbl_profile(profile_id)
        - On update cascade, on delete no action
    - Foreign key fk_software_id references tbl_software(software_id)
        - On update cascade, on delete no action
- **tbl_uses_provisioners** (profile_id, provisioner_id)
    - Primary key profile_id, provisioner_id

    - Foreign key fk_profile_id references tbl_profile(profile_id)
        - On update cascade, on delete no action
    - Foreign key fk_provisioner_id references tbl_provisioner(provisioner_id)
        - On update cascade, on delete no action
- **tbl_uses_providers** (profile_id, provider_id)
    - Primary key profile_id, provider_id
    - Foreign key fk_profile_id references tbl_profile(profile_id)
        - On update cascade, on delete no action
    - Foreign key fk_provider_id references tbl_provider(provider_id)
        - On update cascade, on delete no action

tbl_members **follow** zero or many Vendors, Software, Profiles, JobRoles, Domains, Providers, and Provisioners. These zero-to-many relationships require new relations.

- **tbl_likes_vendor** (member_id, vendor_id)
    - Primary key member_id, vendor_id, fk_member_role_id
    - Foreign key fk_vendor_id references tbl_vendor(vendor_id)
        - On update cascade, on delete no action
    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_member_role_id references UserRole (fk_member_role_id)
- **tbl_likes_software** (member_id, software_id)
    - Primary key member_id, software_id, fk_member_role_id
    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_software_id references tbl_software(software_id)
        - On update cascade, on delete no action
- **tbl_likes_profile** (member_id, profile_id)
    - Primary key member_id, profile_id, fk_member_role_id
    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_profile_id references tbl_profile(profile_id)
        - On update cascade, on delete no action
- **tbl_likes_jobrole** (member_id, jobrole_id)
    - Primary key member_id, jobrole_id, member_role_id
    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_jobrole_id references tbl_jobrole(jobrole_id)
        - On update cascade, on delete no action
- **tbl_likes_domain** (member_id, domain_id)
    - Primary key jobrole_id, member_id, fk_member_role_id
    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_domain_id references Domain (domain_id)
        - On update cascade, on delete no action
- **tbl_likes_provisioner** (member_id, provisioner_id)
    - Primary key member_id, provisioner_id, fk_member_role_id

    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_provisioner_id references tbl_provisioner(provisioner_id)
        - On update cascade, on delete no action
- **tbl_likes_provider** (provider_id, member_id)
    - Primary key provider_id, member_id
    - Foreign key fk_member_id references tbl_member (member_id)
        - On update cascade, on delete no action
    - Foreign key fk_provider_id references tbl_provider(provider_id)
        - On update cascade, on delete no action

Provisioners **support** one or many Providers. This mandatory one-to-many relationship requires a new relation.

- **tbl_supports_provider** (provisioner_id, provider_id)
    - Primary key provisioner_id, provider_id
    - Foreign key fk_provisioner_id references tbl_provisioner(provisioner_id)
        - On update cascade, on delete no action
    - Foreign key fk_provisioner_id references tbl_provisioner(provisioner_id)
        - On update cascade, on delete no action

Normalised Tables

Database normalisation is a iterative and systemic technique of organising the data in the database.. At its simplest forms, it's the process of applying common sense.

There are no multivalued attributes, so that's third normal form. Attributes are independent so that's forth normal form. No join dependencies achieves fifth normal form. Associations are preserved by including foreign keys within the primary key of every relationship table.

Table Design

This section is a full description of each table identifying primary, secondary, alternate and foreign keys, field format, size, Null (or Non-Null), and integrity. As a general rule, Alternate Keys are unique and non-Null. In future cloud, container, and hypervisor tables are needed, but for now Null Foreign Keys are allowed in Provider Table.

Table	Field	T	S	Nulls		Defaults Constraints	Index	Set of Values	Description
tbl_member_role									
	member_role_id	I	8	NN		Unique	PK	1..n	tbl_member_role Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Role name
tbl_member									
	member_id	I	8	NN		Unique	PK	1..n	tbl_member Id
	f_name	C	24	NN		~~Unique~~	AK	a-z,A-Z,0-n , punc	First Name
	l_name	C	45	N				a-z,A-Z,0-n , punc	Last Name
	member_role_id	I	8	NN		Unique	FK		
	api_version	C	8	NN				1..n, .	Devstack API
	Profiles, JobRoles, Domains, Software, Providers, Provisioners								Multivalued
tbl_software_role									
	software_role_id	I	8	NN		Unique	PK	1..n	Software_role Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Role name
tbl_software									
	software_id	I	8	NN		Unique	PK	1..n	Software Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Software Name

	software_role_id	I	8	NN		Unique	FK		
	api_version	C	8	NN				1..n	Devstack API
	Vendors, Profiles, tbl_members							Multivalued	
tbl_vendor_role									
	vendor_role_id	I	8	NN		Unique	PK	1..n	Vendor_role Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Role name
tbl_vendor									
	vendor_id	I	8	NN		Unique	PK	1..n	Vendor Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Vendor Name
	vendor_role_id	I	8	NN		Unique	FK		
	api_version	C	8	NN				1..n	Devstack API
tbl_profile_role									
	profile_id	I	8	NN		Unique	PK	1..n	Profile role Id
	name	C	128	NN		Unique	AK	a-z,A-Z,0-n , punc	Profile role Name
tbl_profile									
	profile_id	I	8	NN		Unique	PK	1..n	Profile Id
	name	C	128	NN		Unique	AK	a-z,A-Z,0-n , punc	Profile Name
	url	C	256	NN				a-z,A-Z,0-n , punc	Profile URL
	jobrole_id	I	8	NN					Job Role ID
	domain_id	I	8	NN					Domain ID
	api_version	C	8	NN				1..n, .	Devstack API
	Providers, Software, Provisioners, tbl_members							Multivalued	
tbl_jobrole									
	jobrole_id	I	8	NN		Unique	PK	1..n	JobRole Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	JobRole Name
	type	C	8	N				a-z,Z-Z,0-n, Null	JobRole Type
	api_version	C	8	N				1..n, .	Devstack API
	Profiles, Domains, tbl_members							Multivalued	
tbl_domain									
	domain_id	I	8	NN		Unique	PK	1..n	Domain Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Domain name
	type	C	8	N				a-z,Z-Z,0-n, Null	Domain type
	api_version	C	8	~~NN~~				1..n, .	Devstack API
	JobRoles, tbl_members							Multivalued	
tbl_provisioner_role									
	provisioner_role_id	I	8	NN		Unique	PK	1..n	Profile role Id
	name	C	128	NN		Unique	AK	a-z,A-Z,0-n , punc	Profile role Name
tbl_provisioner									
	provisioner_id	I	8	NN		Unique	PK	1..n	Provisioner Id
	name	C	45	NN		Unique	AK	a-z,A-Z,0-n , punc	Provisioner name
	type	C	8	N				a-z,Z-Z,0-n, Null	Provisioner type
	api_version	C	8	~~NN~~				1..n, .	Devstack API
	Providers, tbl_members							Multivalued	
tbl_provider_role									
	provider_role_id	I	8	NN		Unique	PK	1..n	Profile role Id
	name	C	128	NN		Unique	AK	a-z,A-Z,0-n , punc	Profile role Name
tbl_provider									
	provider_id	C	8	NN		Unique	PK	1..n	Provider Id
	name	C	48	NN		Unique	AK	a-z,A-Z,0-n , punc	Provider Name
	type	C	8	NN				a-z,Z-Z,0-n, Null	Provider type
	api_version	C	8	NN				1..n, .	Devstack API
	Provisioners, Profiles, tbl_members, CloudId, HypervisorId, ContainerId, OsId							Multivalued	
tbl_vends_software									
	vendor_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Vendor Id
	software_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Software Id
tbl_vends_provider									
	vendor_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Vendor Id
	provider_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provisioner Id
tbl_vends_provisioner									
	vendor_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Vendor Id
	provisioner_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provisioner Id
tbl_uses_software									
	profile_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Profile Id
	software_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Software Id
tbl_uses_provisioner									
	profile_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Profile Id
	provisioner_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provisioner Id
tbl_uses_provider									
	profile_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Profile Id

	provider_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provisioner Idt
tbl_likes_vendor									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	vendor_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Vendor Id
tbl_likes_software									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	software_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Software Id
tbl_likes_profile									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	profile_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Profile Id
tlb_likes_jobrole									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	jobrole_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	JobRole Id
tbl_likes_domain									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	domain_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Domain Id
tbl_likes_provisioner									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	provisioner_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provisioner Id
tbl_likes_provider									
	member_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	tbl_member Id
	provider_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provider Id
tbl_supports_provider									
	provisioner_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provisioner Id
	provider_id	I	8	NN		UQ, onupdate cascade, on delete no-action	PK, FK	1..n	Provider Id

Physical Implementation

The physcial implementation of my database involves converting converting my logical database design to a physical implementation. MySQL will be the implementation choice. My approach uses the Model Design feature of MysqlWorkbench product and manual editing of the resulting scripts until a working soution is completed.

Table Implementtion

My '**Table Design'** is translated into a physical database implementation using an iterative process of reading the design and implementing in MySQL. During the exercise some changes were either desirable, necessary, or both. This document has been updated accordingly.

Here is a detailed list of the design revisions-

1. All tables are now prefixed by '**tbl_**' and all foreign keys by the '**fk_**' prefix following a convention which seems a good practice to me going forward.

2. Furthermore, all foreign keys are now prefixed '**fk_<table_name>**' because, in my experience, MySQL refused to accept duplicate foreign key names.

3. All tables names are now **lowercase**-**underscored** as best practice convention.

4. All fields named`**type**` to `**desc**` because this was design error.

5. All Relations with foreign key fields with '**on update CASCADE'** constraints were changed to '**NO ACTION'** because MySQL does not allow updates from relations to cascade upwards through foreign keys.

6. All `**api_version**` columns were removed from "Relations" tables - versioning a relationship between entities is unnecesary.

7. A design change was required. In order to populate ***sane data*** across domain, jobrole, and profile with flexibility, this design change was needed-

SQL	Reason
DROP TABLE IF EXISTS tbl_holds_jobrole	Table not needed – keep jobrole independent of domain.

SQL	Reason
DROP TABLE IF EXISTS tbl_holds_profile	Table not needed – keep profile indepedent of domain.
ALTER TABLE tbl_jobrole IF EXISTS DROP COLUMN domain_id	Jobrole entity is independent of domain.
ALTER TABLE tbl_profile IF EXISTS DROP COLUMN domain_id	profile entity is independent of domain.

Users and Privledges

The following users were created for the reasons given-

1. **adminguy** – alias for root user, with **grant all** privledges on all DB objects.

2. **juniordba** – junior dba master user, with DELETE, GRANT OPTION, INSERT, SELECT, UPDATE, CREATE privledges on all tables. Junior DBA's have restricted permissions to strengthen data protection.

3. **seniordba** – dba, with **grant all** privledges on all DB objects. Senior DBA's need enchanced permissions to manage the database.

4. **vendorguy** – business user to manage vendor related data, with CREATE, GRANT OPTION, DELETE, INSERT, SELECT, UPDATE privledges only on **tbl_vendor*, tbl_vends_*** tables.

5. **memberguy** – business user to manage members, with CREATE, GRANT OPTION, DELETE, INSERT, SELECT, UPDATE privledges only on **tbl_member*** tables.

6. **softwareguy** – business user to manage software related data, with CREATE, GRANT OPTION, DELETE, INSERT, SELECT, UPDATE privledges only on **tbl_software, tbl_vends_*** tables.

7. **domainguy** – business user to manage domain, jobrole, and profile related data, with CREATE, GRANT OPTION, DELETE, INSERT, SELECT, UPDATE privledges only on **tbl_jobrole, tbl_domain, tbl_profile*, tbl_holds*, tbl_uses*** tables.

8. **solutionguy** – most powerful business user of the database, with CREATE, GRANT OPTION, DELETE, INSERT, SELECT, UPDATE privledges only on **tbl_vendor*, tbl_vends_*, tbl_software, tbl_jobrole, tbl_domain, tbl_profile*, tbl_holds*, tbl_uses*** tables.

9. **webapp** – system user for authentication and authorization of WebApps. Has CISUD access to all tables.

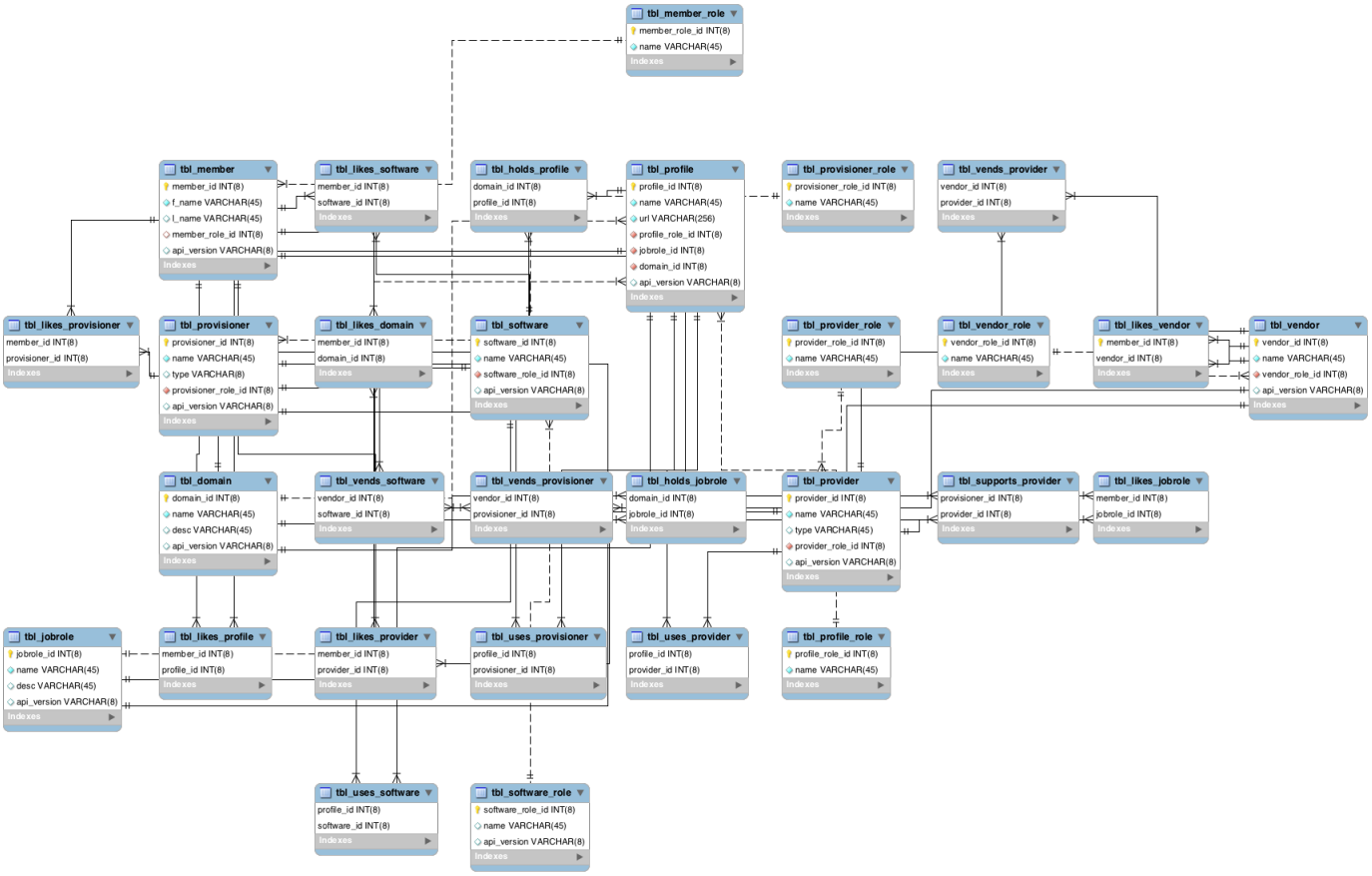
Frequent Queries

The tables shows some frequently used queries on this database, with explanation.

Query	Common Usage
show tables;	List all tables.
select name from tbl_domain;	List available domains.
select concat(f_name, ' ', l_name) as Name from tbl_member where f_name like "%Noe%";	List members with firstname like "Noe".
select tbl_domain.name as "Domain", tbl_jobrole.name as "Job Role", tbl_profile.name as "Profile Name" from tbl_profile join tbl_domain join tbl_jobrole where (tbl_profile.domain_id = tbl_domain.domain_id and tbl_profile.jobrole_id = tbl_jobrole.jobrole_id);	List profile names in terms of associated domain and jobroles. **Domain/JobRole/ProfileName**
select tbl_domain.name as "Domain Name", count(tbl_profile.name) as "Number of Profiles" from tbl_domain join tbl_profile on tbl_domain.domain_id = tbl_profile.domain_id where tbl_domain.name like "%Big%" group by tbl_profile.domain_id;	List profiles associated with "Big"(data) domains.
select * from tbl_profile join tbl_uses_software on tbl_profile.profile_id = tbl_uses_software.profile_id join tbl_software on tbl_uses_software.software_id = tbl_software.software_id where tbl_software.name like "%mysql%";	Find profiles using 'mysql' software.
select tbl_profile.name as "Profile Name", tbl_provisioner.name as "Provisioner Name"	List profiles using 'salt' provisioner.

Query	Common Usage
from tbl_profile join tbl_uses_provisioner on tbl_profile.profile_id = tbl_uses_provisioner.profile_id join tbl_provisioner on tbl_uses_provisioner.provisioner_id = tbl_provisioner.provisioner_id where tbl_provisioner.name like "%salt%";	
select DISTINCT tbl_provisioner.name as "Provisioner's supporting the Vagrant Provider" from tbl_provider join tbl_supports_provider on tbl_provider.provider_id = tbl_supports_provider.provider_id join tbl_provisioner on tbl_provider.provider_id = tbl_supports_provider.provider_id where tbl_provider.name like "%vagrant%";	List Provisioners supporting ‘vagrant’ provider.
select * from tbl_software; insert into tbl_software (software_id, name, software_role_id, api_version) values (8, "wordpress", 1, "1"), (9, "redis", 1, "1"), (10, "nginx", 1, "1"), (11, "nomad", 1, "1"), (12, "logstash", 1, "1"); select * from tbl_software;	`softwareguy` updates software inventory.
UPDATE tbl_software SET name = "Redis" WHERE name = "redis";	Capitalize a software name.
select * from tbl_profile where domain_id = (select domain_id from tbl_domain where name like "%iot%");	SubQuery example. List profiles in domains described as "IoT-related".

Final EER Diagram



UML

UML diagrams were written in PlantUML using Eclipse Plugin. Here is the source code.

/ER Model **/**

```
@startuml
skinparam  {
  handwritten false
  ArrowColor DeepSkyBlue
  backgroundcolor #LightBlue
  shadowing false
  packagebackgroundcolor #aqua
}

package "devstacks.io" {

    entity "tbl_member_role" as memberrole {
        --attributes--
        [fk_member_role_id]
        [Name]
        [ApiVersion]
    }
    entity "tbl_software_role" as softwarerole {
        --attributes--
        [SwRoleId]
        [Name]
        [ApiVersion]
    }
    entity "tbl_vendor_role" as vendorrole {
        --attributes--
        [tbl_vendor_roleId]
        [Name]
        [ApiVersion]
    }
    entity "tbl_member" as member {
        --attributes--
        [member_id]
        [fName]
        [lName]
        [ApiVersion]
    }
    entity "Vendor" as vendor {
        --attributes--
        [vendor_id]
        [Name]
        [ApiVersion]
    }
    entity "Software" as software {
        --attributes --
        [software_id] {PK}
        [Name] {AK}
        [ApiVersion]
    }
    entity "Profile" as profile {
        --attributes--
        [profile_id] {PK}
        [Name] {AK}
        [Url]
        [ApiVersion]
    }
    entity "JobRole" as jobrole {
          --attributes--
          [jobrole_id] {PK}
```

/Architectural Diagram **/**

```
/***
 *
 **/
@startuml
skinparam  {
  handwritten true
  ArrowColor DeepSkyBlue
  backgroundcolor #LightBlue
  packagebackgroundcolor #aqua
}

entity devstacks.io <<Devstacks Hub>>
artifact Devstackfile <<disposable artifact>> #FF9CE4
entity "[Domain/Role/Profile(Provider, Provisioner)]\n"
<<Base and Contributed>>
entity "Infrastructure\n" <<Third Party>>

entity "ansible.com\n" <<infra-as-code>>
entity "x.com\n" <<code>>
entity "docker.com\n" <<containers>>
entity "saltstack.com\n" <<infra-as-code>>
entity "vagrant.com\n" <<VM images>>
entity "citrix.com\n" <<hypervisor>>
entity "vmware.com\n" <<hypervisor>>
entity "virtualbox.com\n" <<hypervisor>>

entity "Host" <<Operating System (Darwin, Linux)>>
entity devstackSW <<client software>>
<<Provisioned>>
entity providerSW <<client software>>
entity provisionerSW <<client software>>

together "usecase" {
          usecase "Auto-provision my software stack
profile\naccording the role and domain
interests\nchoosen from devstacks.io" #29FF5A
          actor Developer #29FF5A
}
package "devstacks.io" #FF9CE4 {
          package "[Domain/Role/Profile(Provider,
Provisioner)]\n" #e6fff3 {
                    Database "Domain" #29FF5A{
                              [Machine Learning] #409DE8
                              [IoT] #409DE8
                              [Artificial Intelligence]
#409DE8
                              [Blockchain] #409DE8
                              [FullStack] #409DE8
                              [Platforms] #409DE8
                              [Storage] #409DE8
                              [Fun] #409DE8
                              [..etc..] #409DE8
                    }
                    Database "JobRole" #29FF5A{
                              [Engineer] #409DE8
                              [Architect] #409DE8
                              [Scientist] #409DE8
                              [Researcher] #409DE8
                              [Programmer] #409DE8
```

```
        [Name] {AK}
        [Type]
        [ApiVersion]
    }
entity "Domain" as domain {
        --attributes--
        [domain_id] {PK}
        [Name] {AK}
        [Type]
        [ApiVersion]
    }
entity "Provisioner" as provisioner {
    --attributes--
    [provisioner_id] {PK}
    [Name] {AK}
    [Type]
    [ApiVersion]
}
entity "Provider" as provider {
    --attributes--
    [provider_id]
    [Name]
    [Type]
        [ApiVersion]
}
entity "CloudProvider" as cloud {
    --attributes--
    [cloudId] {FK}
}
entity "ContainerProvider" as container {
    --attributes--
    [containerId] {FK}
}
entity "HypervisorProvider" as hypervisor {
    --attributes--
    [hypervisorId] {FK}
}
entity "OsProvider" as os {
    --attributes--
    [software_id] {FK}
}
provider <|-down- cloud: {Mandatory, Or}
provider <|-down- hypervisor: {Mandator, Or}
provider <|-down- container: {Mandatory, Or}
provider <|-down- os: {Mandatory, Or}
software -[hidden]- provisioner

vendor "0" *-- "*" software: vends >
software "1" *-- "*" vendor: vended by >

vendor "0" *-right- "*" provider: vends >
provider "1" *- "*" vendor: vended by >

vendor "0" *-- "*" provisioner: vends >
provisioner "1" *-- "*" vendor: vended by >

domain "1" *-- "*" jobrole: holds >
jobrole "1" *-- "*" domain: held by >

jobrole "1" *-- "*" profile: holds >
profile "1" *-- "*" jobrole: held by >

profile "1" *-- "*" software: uses >
software "0" *-- "*" profile: used by >

profile "1" *-- "*" provisioner: uses >
provisioner "0" *-left- "1" profile: used by >
```

```
                [.etc.]  #409DE8
            }
            Database "Profile" #29FF5A{

        [01011011010011000110011001010 11]
#FF9CE4
            }
            Database "tbl_member" #29FF5A{
                [member]
            }
            JobRole -[hidden]-- Profile
            tbl_member -[hidden]- Profile
        }
        package "Infrastructure\n" #e6fff3 {
            Database "Provider" #e6fff3 {
                Database "Hypervisor" {
                    [VBox]
                    [Kvm]
                }
                Database "Container" {
                    [Lxc]
                    [Docker]
                }
                Database "Cloud" {
                    [aws]
                }
            }
            Database "Provisioner" #e6fff3{
                [Salt]
                [Ansible]
                [terraform]
                [chef]
                [shell]
                [Vagrant]
            }
            Database "Software" #e6fff3 {
                [software] #e6fff3
            }
            Database "Vendor" #e6fff3 {
                [software]
            }
        }
        Provider -right- Provisioner
        Provider -down- Software
        Software -down- Vendor
        Provider -down- Vendor
        Provisioner -down- Vendor
}

package "Host" #FF9CE4 {

        interface providerAPI
        interface provisionerAPI
        interface devstackAPI

        package "devstackSW" #e6fff3 {
            [Devstack] #FF9CE4
        }
        package "provisionerSW"  #e6fff3 {
            [PKG or Ansible or Salt or Terraform
or Chef or X] -down-( provisionerAPI
        }
        package "providerSW" #e6fff3 {
            [ProviderTypeSw] --( providerAPI
        }

        Devstack --( provisionerAPI
        provisionerSW -down- devstackAPI
```

```
    profile "1" *-- "*" provider: uses >
    provider "0" *-left- "1" profile: used by >

    member "1" *-- "*" vendor: like >
    vendor "0" *-- "*" member: liked by >

    member "1" *-- "*" software: like >
    software "0" *-- "*" member: liked by >

    member "1" *-- "*" profile: like >
    profile "0" *-- "*" member: liked by >

    member "1" *-- "*" jobrole: like >
    jobrole "0" *-- "*" member: liked by >

    member "1" *-- "*" domain: like >
    domain "0" *-- "*" member: liked by >

    member "1" *-- "*" provisioner: like >
    provisioner "0" *-- "*" member: liked by >

    member "1" *-- "*" provider: like >
    provider "0" *-- "*" member: liked by >

    provisioner "1" *-- "*" provider: supports >
    provider "1" *-- "*" provisioner: supported by >

    member "1" *-- "*" memberrole: has >
    memberrole "0" *-- "*" member: has >
    vendor "1" *-- "*" vendorrole: has >
    vendorrole "0" *-- "*" vendor: has >
    software "1" *-- "*" softwarerole: has >
    softwarerole "0" *-- "*" software: has >
}
@enduml
```

```
        Devstackfile -down- Developer
}
actor Developer
Developer -up-devstackAPI
Devstack -down- devstackAPI
devstackSW -left- providerSW

Vendor --down-- Host
providerSW -right-( [devstackAPI]


together {
        component "ansible.com\n" #e6fff3 {
                component "ansible"
                [ansible] -( [provisionerAPI]
        }
        component "x.com\n" #e6fff3 {
                component formulas
                formulas -up-( [provisionerAPI]
        }
        component "saltstack.com\n" #e6fff3 {
                component salt
                salt -up-( [provisionerAPI]
        }
        component "docker.com\n" #e6fff3 {
                component docker
                docker -up-( [providerAPI]
        }
        component "vagrant.com\n" #e6fff3 {
                component vagrant
                vagrant -up--( [providerAPI]
        }
        component "vmware.com\n" #e6fff3 {
                component vmware
                vmware -up--( [providerAPI]
        }
        component "citrix.com\n" #e6fff3 {
                component citrix
                citrix -up--( [providerAPI]
        }
        component "virtualbox.com\n" #e6fff3 {
                component virtualbox
                virtualbox ---( [providerAPI]
        }
}
@enduml
```

Database Security

Data protection involves the prevention of intentional and unintentional read, update, and deletion of records contained within the database, by unauthorized third-parties and database users. For the devstacks database the following are implemented-

- User authentication to ensure database access is authroized.
- Access control to ensure each user's access is managed.

Two types of user are implemented for the devstacks database-

1. user account for each record in the **tbl_members** table.
2. administrator account.