

 main ▾

...

[pulp_ansible](#) / [pulp_ansible](#) / [app](#) / [models.py](#) / [Jump to](#) ▾



jctanner collectionversion search_vector should not include every tag ever (#1277 ...

 History

 10 contributors



419 lines (322 sloc) | 14.1 KB

...

```
1  from logging import getLogger
2
3  from django.conf import settings
4  from django.db import models
5  from django.db.models import UniqueConstraint, Q
6  from django.contrib.postgres import fields as psql_fields
7  from django.contrib.postgres import search as psql_search
8  from django_lifecycle import AFTER_UPDATE, BEFORE_UPDATE, hook
9
10 from pulpcore.plugin.models import (
11     BaseModel,
12     Content,
13     Remote,
14     Repository,
15     RepositoryVersion,
16     Distribution,
17     SigningService,
18     Task,
19     EncryptedTextField,
20 )
21 from .downloaders import AnsibleDownloaderFactory
22
23
24 log = getLogger(__name__)
25
26
27 class Role(Content):
28     """
29     A content type representing a Role.
```

```

30     """
31
32     TYPE = "role"
33
34     namespace = models.CharField(max_length=64)
35     name = models.CharField(max_length=64)
36     version = models.CharField(max_length=128)
37
38     @property
39     def relative_path(self):
40         """
41         Return the relative path of the ContentArtifact.
42         """
43         return self.contentartifact_set.get().relative_path
44
45     class Meta:
46         default_related_name = "%(app_label)s_%(model_name)s"
47         unique_together = ("version", "name", "namespace")
48
49
50 class Collection(BaseModel):
51     """A model representing a Collection."""
52
53     namespace = models.CharField(max_length=64, editable=False)
54     name = models.CharField(max_length=64, editable=False)
55
56     def __str__(self):
57         """Return a representation."""
58         return f"<{self.__class__.__name__}: {self.namespace}.{self.name}>"
59
60     class Meta:
61         unique_together = ("namespace", "name")
62
63
64 class CollectionImport(models.Model):
65     """A model representing a collection import task details."""
66
67     task = models.OneToOneField(
68         Task, on_delete=models.CASCADE, editable=False, related_name="+", primary_key=True
69     )
70     messages = models.JSONField(default=list, editable=False)
71
72     class Meta:
73         ordering = ["task__pulp_created"]
74
75     def add_log_record(self, log_record):
76         """
77         Records a single log message but does not save the CollectionImport object.
78

```

```

79     Args:
80         log_record(logging.LogRecord): The logging record to record on messages.
81
82     """
83     self.messages.append(
84         {"message": log_record.msg, "level": log_record.levelname, "time": log_record.created}
85     )
86
87
88 class Tag(BaseModel):
89     """A model representing a Tag.
90
91     Fields:
92
93         name (models.CharField): The Tag's name.
94     """
95
96     name = models.CharField(max_length=64, unique=True, editable=False)
97
98     def __str__(self):
99         """Returns tag name."""
100         return self.name
101
102
103 class CollectionVersion(Content):
104     """
105     A content type representing a CollectionVersion.
106
107     This model is primarily designed to adhere to the data format for Collection content. That spe
108     is here: https://docs.ansible.com/ansible/devel/dev\_guide/collections\_galaxy\_meta.html
109
110     Fields:
111
112         authors (psql_fields.ArrayField): A list of the CollectionVersion content's authors.
113         contents (models.JSONField): A JSON field with data about the contents.
114         dependencies (models.JSONField): A dict declaring Collections that this collection
115             requires to be installed for it to be usable.
116         description (models.TextField): A short summary description of the collection.
117         docs_blob (models.JSONField): A JSON field holding the various documentation blobs in
118             the collection.
119         manifest (models.JSONField): A JSON field holding MANIFEST.json data.
120         files (models.JSONField): A JSON field holding FILES.json data.
121         documentation (models.CharField): The URL to any online docs.
122         homepage (models.CharField): The URL to the homepage of the collection/project.
123         issues (models.CharField): The URL to the collection issue tracker.
124         license (psql_fields.ArrayField): A list of licenses for content inside of a collection.
125         name (models.CharField): The name of the collection.
126         namespace (models.CharField): The namespace of the collection.
127         repository (models.CharField): The URL of the originating SCM repository.

```

```

128         version (models.CharField): The version of the collection.
129         requires_ansible (models.CharField): The version of Ansible required to use the collection
130         is_highest (models.BooleanField): Indicates that the version is the highest one
131             in the collection. Import and sync workflows update this field, which then
132             triggers the database to [re]build the search_vector.
133
134     Relations:
135
136         collection (models.ForeignKey): Reference to a collection model.
137         tag (models.ManyToManyField): A symmetric reference to the Tag objects.
138     """
139
140     TYPE = "collection_version"
141
142     # Data Fields
143     authors = psql_fields.ArrayField(models.CharField(max_length=64), default=list, editable=False)
144     contents = models.JSONField(default=list, editable=False)
145     dependencies = models.JSONField(default=dict, editable=False)
146     description = models.TextField(default="", blank=True, editable=False)
147     docs_blob = models.JSONField(default=dict, editable=False)
148     manifest = models.JSONField(default=dict, editable=False)
149     files = models.JSONField(default=dict, editable=False)
150     documentation = models.CharField(default="", blank=True, max_length=2000, editable=False)
151     homepage = models.CharField(default="", blank=True, max_length=2000, editable=False)
152     issues = models.CharField(default="", blank=True, max_length=2000, editable=False)
153     license = psql_fields.ArrayField(models.CharField(max_length=32), default=list, editable=False)
154     name = models.CharField(max_length=64, editable=False)
155     namespace = models.CharField(max_length=64, editable=False)
156     repository = models.CharField(default="", blank=True, max_length=2000, editable=False)
157     version = models.CharField(max_length=128, editable=False)
158     requires_ansible = models.CharField(null=True, max_length=255)
159
160     is_highest = models.BooleanField(editable=False, default=False)
161
162     # Foreign Key Fields
163     collection = models.ForeignKey(
164         Collection, on_delete=models.PROTECT, related_name="versions", editable=False
165     )
166     tags = models.ManyToManyField(Tag, editable=False)
167
168     # Search Fields
169     # This field is populated by a trigger setup in the database by
170     # a migration file. The trigger only runs when the table is
171     # updated. CollectionVersions are INSERT'ed into the table, so
172     # the search_vector does not get populated at initial creation
173     # time. In the import or sync workflows, is_highest gets toggled
174     # back and forth, which causes an UPDATE operation and then the
175     # search_vector is built.
176     search_vector = psql_search.SearchVectorField(default="")

```

```

177
178     @property
179     def relative_path(self):
180         """
181         Return the relative path for the ContentArtifact.
182         """
183         return "{namespace}-{name}-{version}.tar.gz".format(
184             namespace=self.namespace, name=self.name, version=self.version
185         )
186
187     def __str__(self):
188         """Return a representation."""
189         return f"<{self.__class__.__name__}: {self.namespace}.{self.name} {self.version}>"
190
191     class Meta:
192         default_related_name = "%(app_label)s_%(model_name)s"
193         unique_together = ("namespace", "name", "version")
194         constraints = [
195             UniqueConstraint(
196                 fields=("collection", "is_highest"),
197                 name="unique_is_highest",
198                 condition=Q(is_highest=True),
199             )
200         ]
201
202
203     class CollectionVersionSignature(Content):
204         """
205         A content type representing a signature that is attached to a content unit.
206
207         Fields:
208             data (models.BinaryField): A signature, base64 encoded. # Not sure if it is base64 encoded
209             digest (models.CharField): A signature sha256 digest.
210             pubkey_fingerprint (models.CharField): A fingerprint of the public key used.
211
212         Relations:
213             signed_collection (models.ForeignKey): A collection version this signature is relevant to.
214             signing_service (models.ForeignKey): An optional signing service used for creation.
215         """
216
217         PROTECTED_FROM_RECLAIM = False
218         TYPE = "collection_signature"
219
220         signed_collection = models.ForeignKey(
221             CollectionVersion, on_delete=models.CASCADE, related_name="signatures"
222         )
223         data = models.TextField()
224         digest = models.CharField(max_length=64)
225         pubkey_fingerprint = models.CharField(max_length=64)

```

```

226 signing_service = models.ForeignKey(
227     SigningService, on_delete=models.SET_NULL, related_name="signatures", null=True
228 )
229
230 class Meta:
231     default_related_name = "%(app_label)s_%(model_name)s"
232     unique_together = ("pubkey_fingerprint", "signed_collection")
233
234
235 class DownloadLog(BaseModel):
236     """
237     A download log for content units by user, IP and org_id.
238     """
239
240     content_unit = models.ForeignKey(
241         Content, on_delete=models.CASCADE, related_name="download_logs"
242     )
243     user = models.ForeignKey(
244         settings.AUTH_USER_MODEL,
245         on_delete=models.CASCADE,
246         null=True,
247         related_name="download_logs",
248     )
249     ip = models.GenericIPAddressField()
250     extra_data = models.JSONField(null=True)
251     user_agent = models.TextField()
252     repository = models.ForeignKey(
253         Repository, on_delete=models.CASCADE, related_name="download_logs"
254     )
255     repository_version = models.ForeignKey(
256         RepositoryVersion, null=True, on_delete=models.SET_NULL, related_name="download_logs"
257     )
258
259
260 class RoleRemote(Remote):
261     """
262     A Remote for Ansible content.
263     """
264
265     TYPE = "role"
266
267     class Meta:
268         default_related_name = "%(app_label)s_%(model_name)s"
269
270
271 class CollectionRemote(Remote):
272     """
273     A Remote for Collection content.
274     """

```

```

275
276     TYPE = "collection"
277
278     requirements_file = models.TextField(null=True)
279     auth_url = models.CharField(null=True, max_length=255)
280     token = EncryptedTextField(null=True)
281     sync_dependencies = models.BooleanField(default=True)
282     signed_only = models.BooleanField(default=False)
283
284     @property
285     def download_factory(self):
286         """
287         Return the DownloaderFactory which can be used to generate asyncio capable downloaders.
288
289         Upon first access, the DownloaderFactory is instantiated and saved internally.
290
291         Plugin writers are expected to override when additional configuration of the
292         DownloaderFactory is needed.
293
294         Returns:
295             DownloadFactory: The instantiated DownloaderFactory to be used by
296                             get_downloader()
297
298         """
299         try:
300             return self._download_factory
301         except AttributeError:
302             self._download_factory = AnsibleDownloaderFactory(self)
303             return self._download_factory
304
305     @hook(
306         AFTER_UPDATE,
307         when_any=["url", "requirements_file", "sync_dependencies", "signed_only"],
308         has_changed=True,
309     )
310     def _reset_repository_last_synced_metadata_time(self):
311         AnsibleRepository.objects.filter(
312             remote_id=self.pk, last_synced_metadata_time__isnull=False
313         ).update(last_synced_metadata_time=None)
314
315     class Meta:
316         default_related_name = "%(app_label)s_%(model_name)s"
317
318
319     class GitRemote(Remote):
320         """
321         A Remote for Collection content hosted in Git repositories.
322         """
323

```

```

324     TYPE = "git"
325
326     metadata_only = models.BooleanField(default=False)
327     git_ref = models.TextField()
328
329     class Meta:
330         default_related_name = "%(app_label)s_%(model_name)s"
331
332
333 class AnsibleCollectionDeprecated(Content):
334     """
335     A model that represents if a Collection is `deprecated` for a given RepositoryVersion.
336     """
337
338     TYPE = "collection_deprecation"
339
340     namespace = models.CharField(max_length=64, editable=False)
341     name = models.CharField(max_length=64, editable=False)
342
343     class Meta:
344         default_related_name = "%(app_label)s_%(model_name)s"
345         unique_together = ("namespace", "name")
346
347
348 class AnsibleRepository(Repository):
349     """
350     Repository for "ansible" content.
351
352     Fields:
353
354         last_synced_metadata_time (models.DateTimeField): Last synced metadata time.
355     """
356
357     TYPE = "ansible"
358     CONTENT_TYPES = [
359         Role,
360         CollectionVersion,
361         AnsibleCollectionDeprecated,
362         CollectionVersionSignature,
363     ]
364     REMOTE_TYPES = [RoleRemote, CollectionRemote]
365
366     last_synced_metadata_time = models.DateTimeField(null=True)
367     gpgkey = models.TextField(null=True)
368
369     class Meta:
370         default_related_name = "%(app_label)s_%(model_name)s"
371
372         permissions = (("modify_ansible_repo_content", "Can modify ansible repository content"),)

```



```

373
374 def finalize_new_version(self, new_version):
375     """Finalize repo version."""
376     removed_collection_versions = new_version.removed(
377         base_version=new_version.base_version
378     ).filter(pulp_type=CollectionVersion.get_pulp_type())
379
380     # Remove any deprecated and signature content associated with the removed collection
381     # versions
382     for version in removed_collection_versions:
383         version = version.cast()
384
385         signatures = new_version.get_content(
386             content_qs=CollectionVersionSignature.objects.filter(signed_collection=version)
387         )
388         new_version.remove_content(signatures)
389
390         other_collection_versions = new_version.get_content(
391             content_qs=CollectionVersion.objects.filter(collection=version.collection)
392         )
393
394         # AnsibleCollectionDeprecated applies to all collection versions in a repository,
395         # so only remove it if there are no more collection versions for the specified
396         # collection present.
397         if not other_collection_versions.exists():
398             deprecations = new_version.get_content(
399                 content_qs=AnsibleCollectionDeprecated.objects.filter(
400                     namespace=version.namespace, name=version.name
401                 )
402             )
403
404             new_version.remove_content(deprecations)
405
406     @hook(BEFORE_UPDATE, when="remote", has_changed=True)
407     def _reset_repository_last_synced_metadata_time(self):
408         self.last_synced_metadata_time = None
409
410
411 class AnsibleDistribution(Distribution):
412     """
413     A Distribution for Ansible content.
414     """
415
416     TYPE = "ansible"
417
418     class Meta:
419         default_related_name = "%(app_label)s_%(model_name)s"

```