Фоновые процессы

Примеры использования

- Массовый импорт данных
- Массовый экспорт данных
- Расчет рейтингов (например, лучшие вопросы)
- Очистка устаревших данных
- И вообще все периодические работы
- Статистика, статистика, статистика

Cron

Задача = команда оболочки (bash) в linux

Crontab

```
MAILTO=pupkin@gmail.com
PATH=/bin:/usr/bin:/home/project/bin
SHELL=/bin/bash
0   */3 *   *   *   root   indexer --rotate --all
0   1   *   *   *   root   backup.sh
0   3   7   *   *   root   big_backup_all.sh
0   *   *   *   *   pupkin /home/ask/manage.py calc_best
*   *   *   *   *   pupkin /home/ask/manage.py send_mail
-   -   -   -   -
|   |   |   |   |
|   |   |   |   --- день недели (0—7) (воскресенье = 0 или 7)
|   |   |   ------- месяц (1—12)
|   |   ----------- день (1—31)
|   --------------- час (0—23)
------------------- минута (0—59)
```

Редактирование Crontab

crontab текущего юзера («свой»)

```
$ EDITOR=gedit crontab -e
```

crontab администратора

```
# EDITOR=gedit crontab -e
```

Management команды

```
# ask/qa/management/commands/warmcache.py
from django.core.management.base import BaseCommand

class Command(BaseCommand):
    help = 'generates best users and tags'

    def add_arguments(self, parser): # optparse
        parser.add_argument('--users', action='store_true',
            dest='user', default=False, help='generate users')
        parser.add_argument('--tags', action='store_true',
            dest='user', default=False, help='generate tags')

    def handle(self, *args, **options):
        if options['users']:
            # to do
        if options['tags']:
            # even more do
```

Кэширование

Виды кэширования в Django

- Кэширование всего сайта (FetchFromCacheMiddleware)
- Кэширование отдельного конроллера (@cache_page)
- Кэширование фрагментов шаблонов
- Низкоуровневое кэширование (API)

Кэширование шаблонов

```
{% load cache %}
{% cache 300 sidebar %}
    .. sidebar ...
{% endcache %}
```

Кэширование с учетом аргументов

```
{% load cache %}
{% cache 300 sidebar request.user.is_authenticated %}
    .. sidebar for logged? in user ..
{% endcache %}
```

Использование API

```
from django.core.cache import cache # default

# В management command
data = User.objects.best()[:10]
data = [ {'id': u.id, 'name': u.name } for d in data ]
cache.set('users_key', data, 3600)

# Во view или template tag:
cache.get('users_key')
```

Механизмы кэширования в Django

- `LocMemCache` - в разделяемой памяти
- `FileBasedCache` - в файловой системе
- `DatabaseCache` - в базе данных
- `MemcachedCache` - в демоне memcached

Настройки кэшей

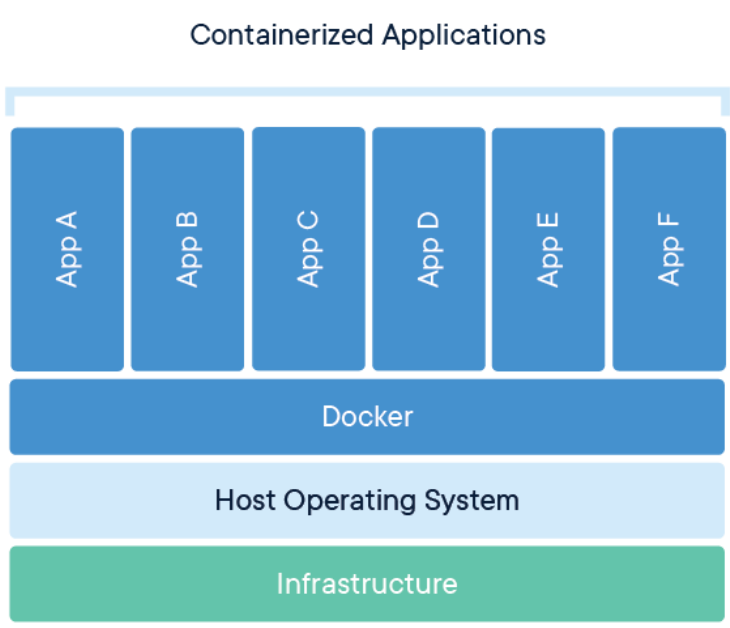
```
# ask/ask/settings.py
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    },
    'filecache': {
        'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
        'LOCATION': '/var/tmp/django_cache',
        'TIMEOUT': 60,
        'OPTIONS': {
            'MAX_ENTRIES': 1000
        }
    },
}
```

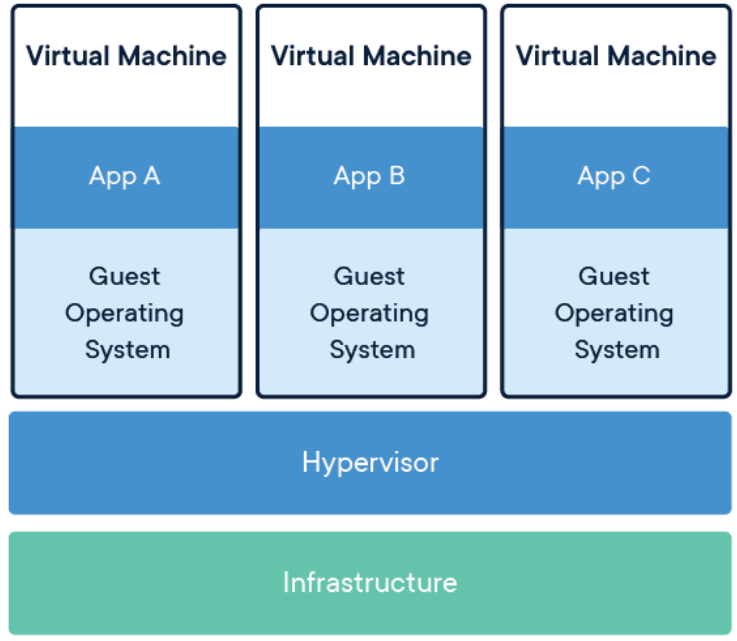
Django REST

```python
from django.urls import path, include
from django.contrib.auth.models import User
from rest_framework import routers, serializers, viewsets

# Serializers define the API representation.
class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ['url', 'username', 'email', 'is_staff']

# ModelViewSet provides .list() .retrieve() .create() .update() .partial_update() .destroy().
class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer

# Routers provide an easy way of automatically determining the URL conf.
router = routers.DefaultRouter()
router.register(r'users', UserViewSet)

# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework'))
]
```

Docker

Container vs Virtual Machine





Kubernetes

Kubernetes architecture