“Fabric”ating RESTful APIs for Linux

Pycon Canada 2016

sachin.agarwal@td.com
@sachinkagarwal
{Gmail|Twitter|LinkedIn|Slack}

IT Provisioning

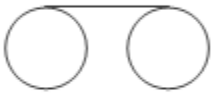
Monitoring

Security/Certificates

User management

Storage

Compute



Backups

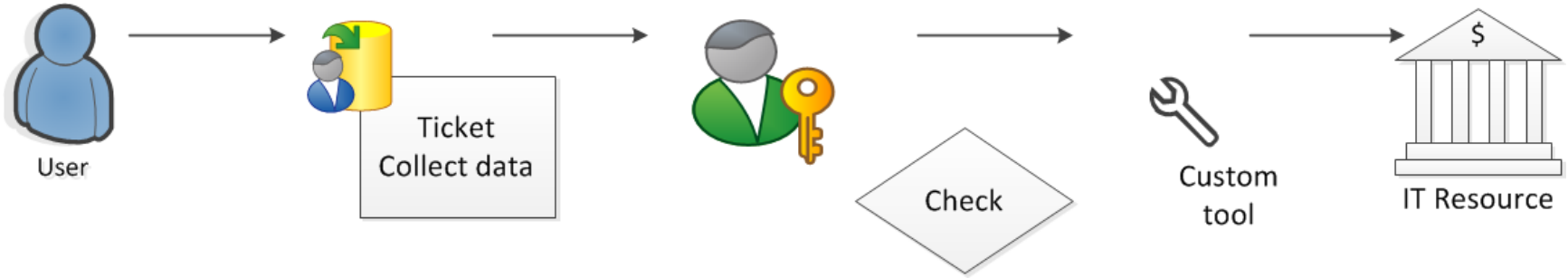
Firewalls

Network Switches

Load Balancers

(Subset)

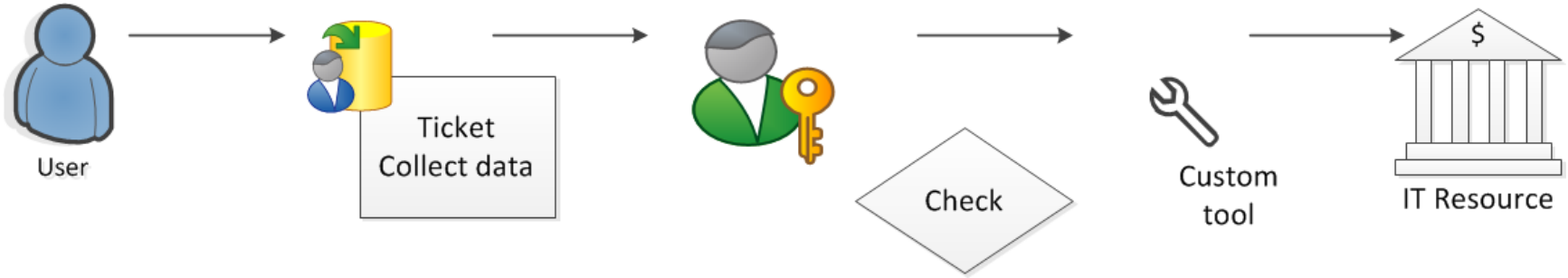
IT Provisioning: The Old Way



The human element

- Does not scale - bottlenecks
- Inconsistencies

Conventional IT Provisioning



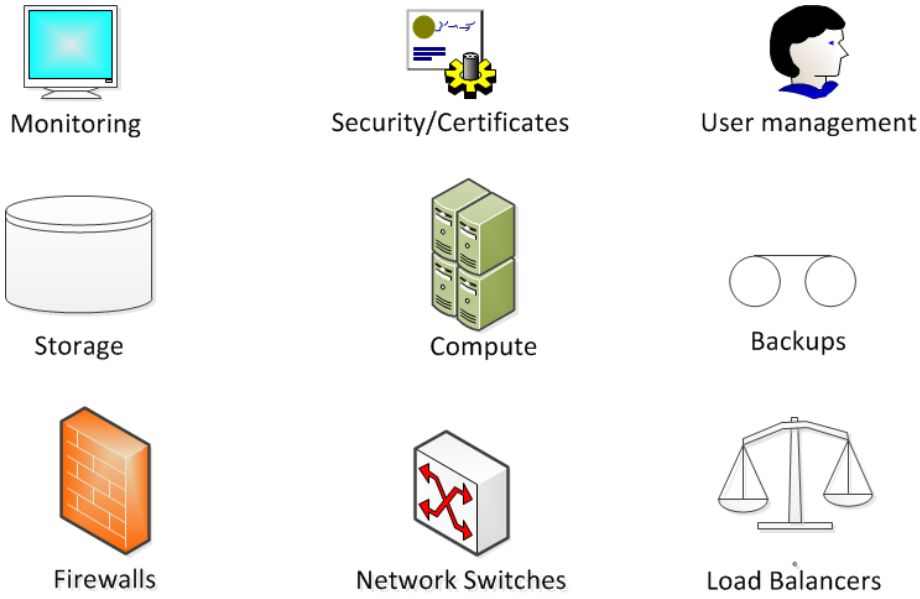
Anti-Devops automation

- Slow
- Violates DRY
- (Repeat process every time)

Enterprise IT: The API Gap

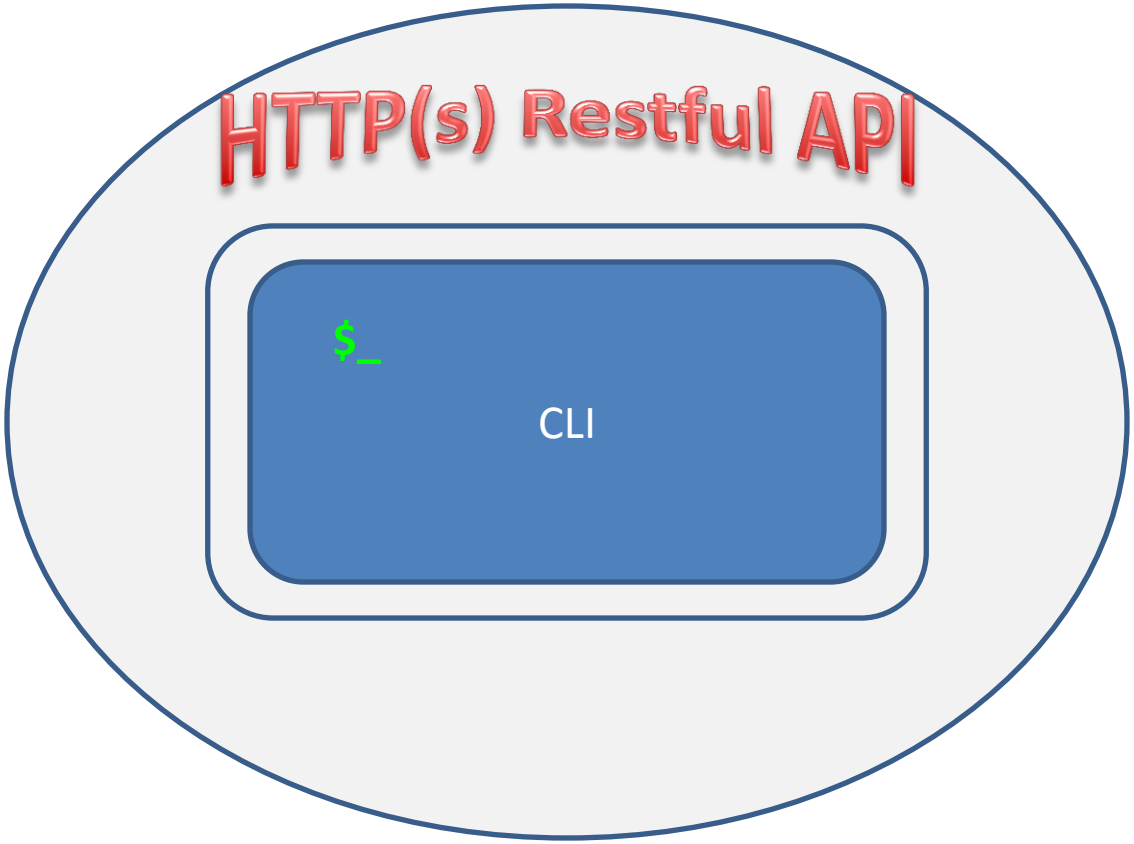
Challenges

- Non-existent or disparate APIs
- Custom CLI* functionality richer than available API functionality
- Reluctance to throw away legacy scripted automation
- Automation vs. control

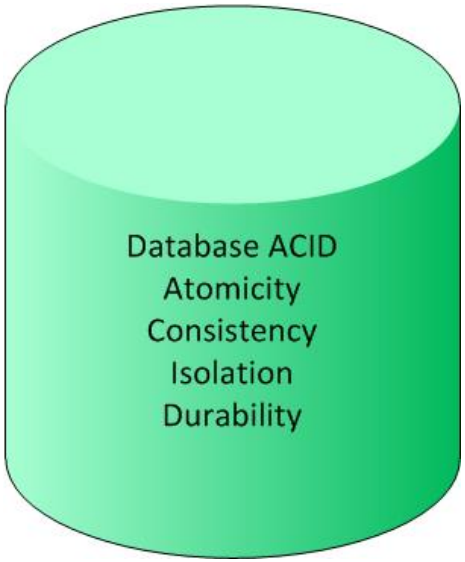


*Command line interface

Build Your Own API



Challenges: more than a Database



VS.


Monitoring


Security/Certificates


User management


Storage


Compute


Backups


Firewalls


Network Switches


Load Balancers

IT infrastructure resources are not databases

NFS Provisioning Vignette
Hootie

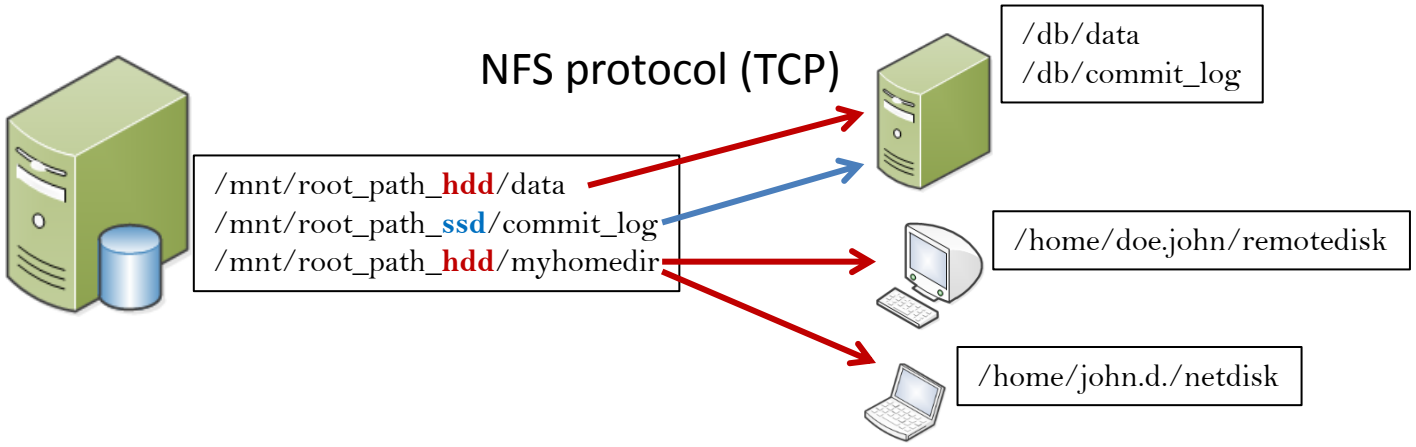
Hootie-all-in-one

git clone https://github.com/sachinkagarwal/hootie.git
cd /hootie/infra
vagrant up

(Refer to the Readme)

NFS* Provisioning

Export a directory over the network as a "volume"



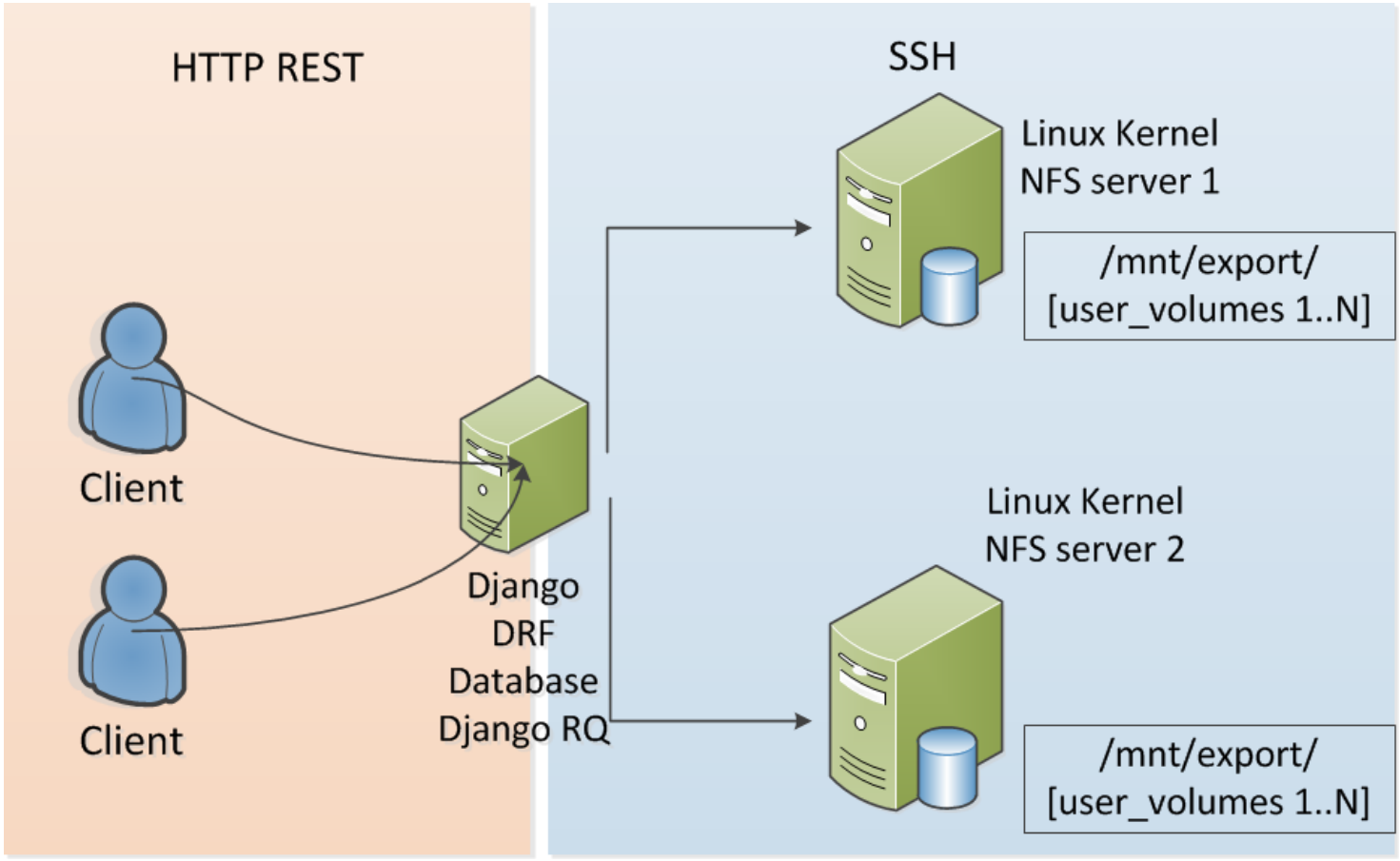
To-do

- Users "provision" volumes on an NFS server via HTTP RESTful APIs (C.R.U.D.)
- User quotas, permissions, capacity tracking

*Network file system

Vignette: NFS Provisioning

Users provision NFS Volumes on Linux servers via an HTTP-RESTful API



Choices

Why Fabric?

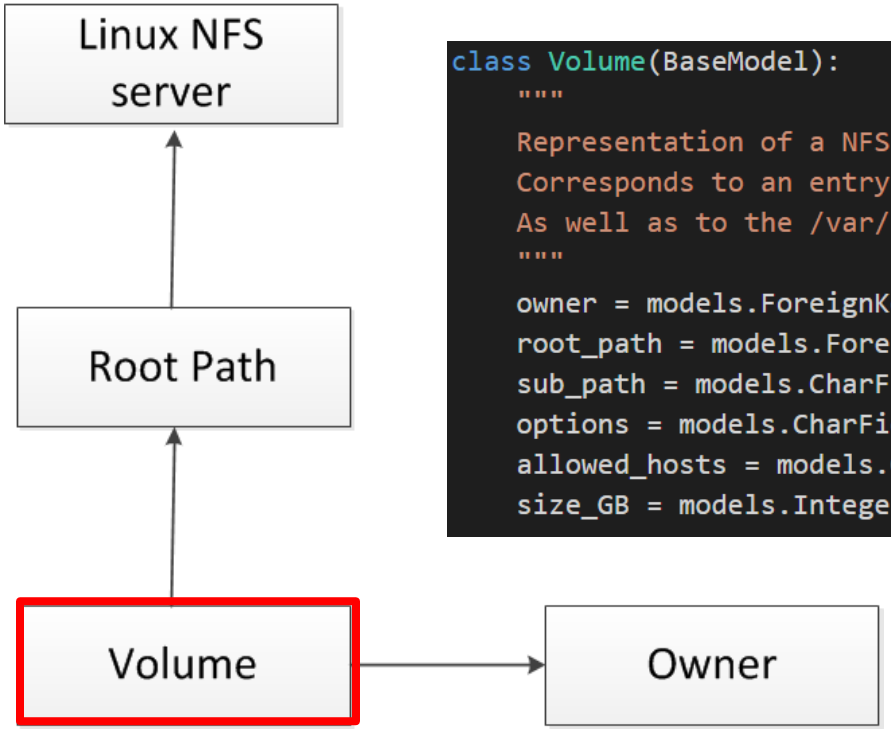
- Agentless
- Widely available protocol (SSH)
- Python integration

Why Django?

- Rich ORM
- Django Rest Framework
- Admin interface
- Django RQs

Resources as Entities

Map a Volume to a Model in the ORM




```python
class Volume(BaseModel):
    """
    Representation of a NFS volume.
    Corresponds to an entry in the /etc/exports file on the NFS server
    As well as to the /var/lib/etab entries
    """
    owner = models.ForeignKey(User, related_name = 'volumes')
    root_path = models.ForeignKey('RootPath')
    sub_path = models.CharField(max_length=200)
    options = models.CharField(max_length=200) #NFS options
    allowed_hosts = models.CharField(max_length=200,default="*") #NFS allowed_hosts
    size_GB = models.IntegerField(default=1, validators=[MinValueValidator(1)])
```

Model Pre-Create Actions

Add custom model save() and delete() methods to configure resources

```
#Override the .save() method of the Volume Model
def save(self, *args, **kwargs):
    """
    Overriding save to create/re-export
    a volume
    """
    execute(CreateOrExport,self.root_path.path,self.sub_path,
            self.allowed_hosts,self.options,
            hosts = [self.root_path.nfs_server.dns_name])
    #Insert test here (on box) to confirm proper provisioning
    #before DB update
    super(Volume,self).save(*args,**kwargs)
```

Bonus: The custom methods work in any view (including Django Admin)

Capacity and Quotas

- Capacity (resource)

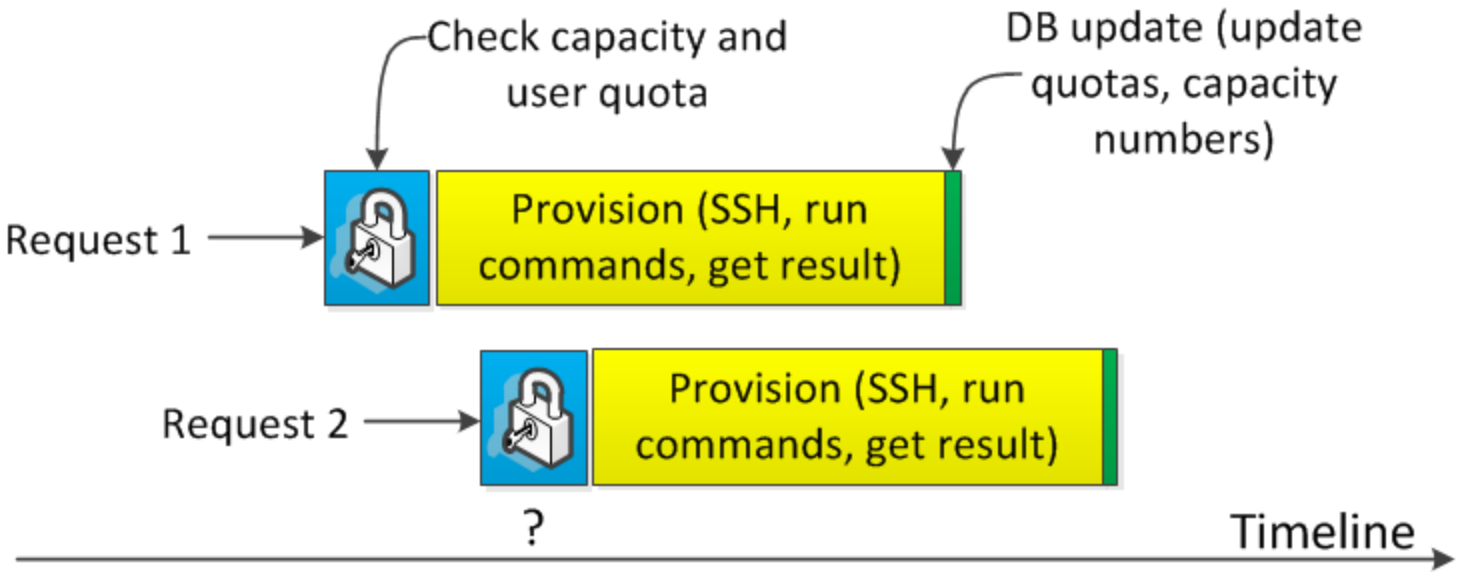
```
class RootPath(BaseModel):
    """
    The path on which NFS volumes will be created.
    There could be multiple such paths per nfs_server.
    """
    nfs_server = models.ForeignKey('NFSServer')
    path = models.CharField(max_length=200)
    capacity_GB = models.IntegerField(default=1, validators=[MinValueValidator(1)])
```

- Quota (user)

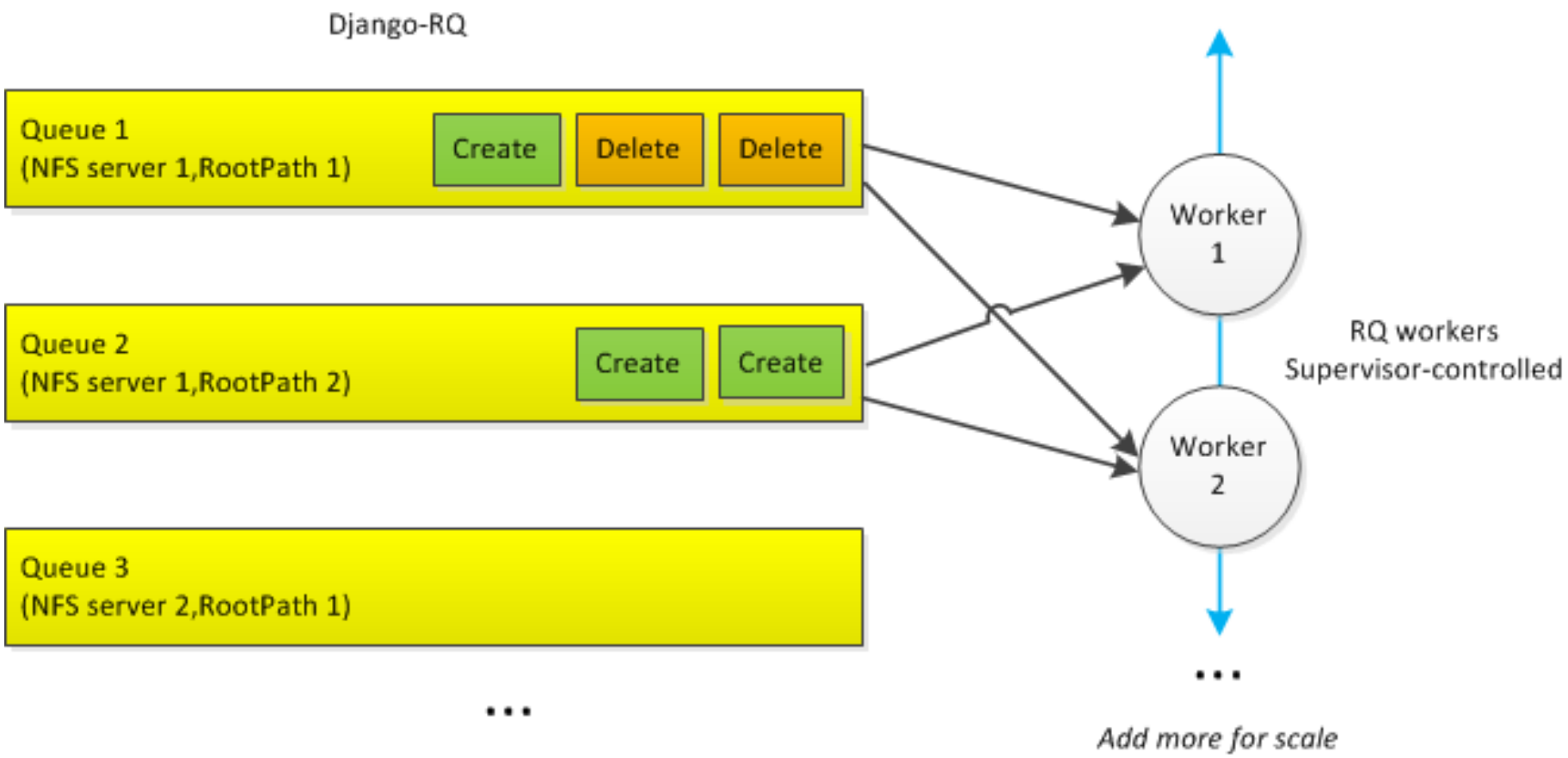
```
#Extend django.contrib.auth.models.User
class Profile(BaseModel):
    user = models.OneToOneField(User,unique=True,primary_key=True)
    quota_GB = models.IntegerField(default=0, validators=[MinValueValidator(0)])
    locked = models.BooleanField(default=False)
```

Serialization

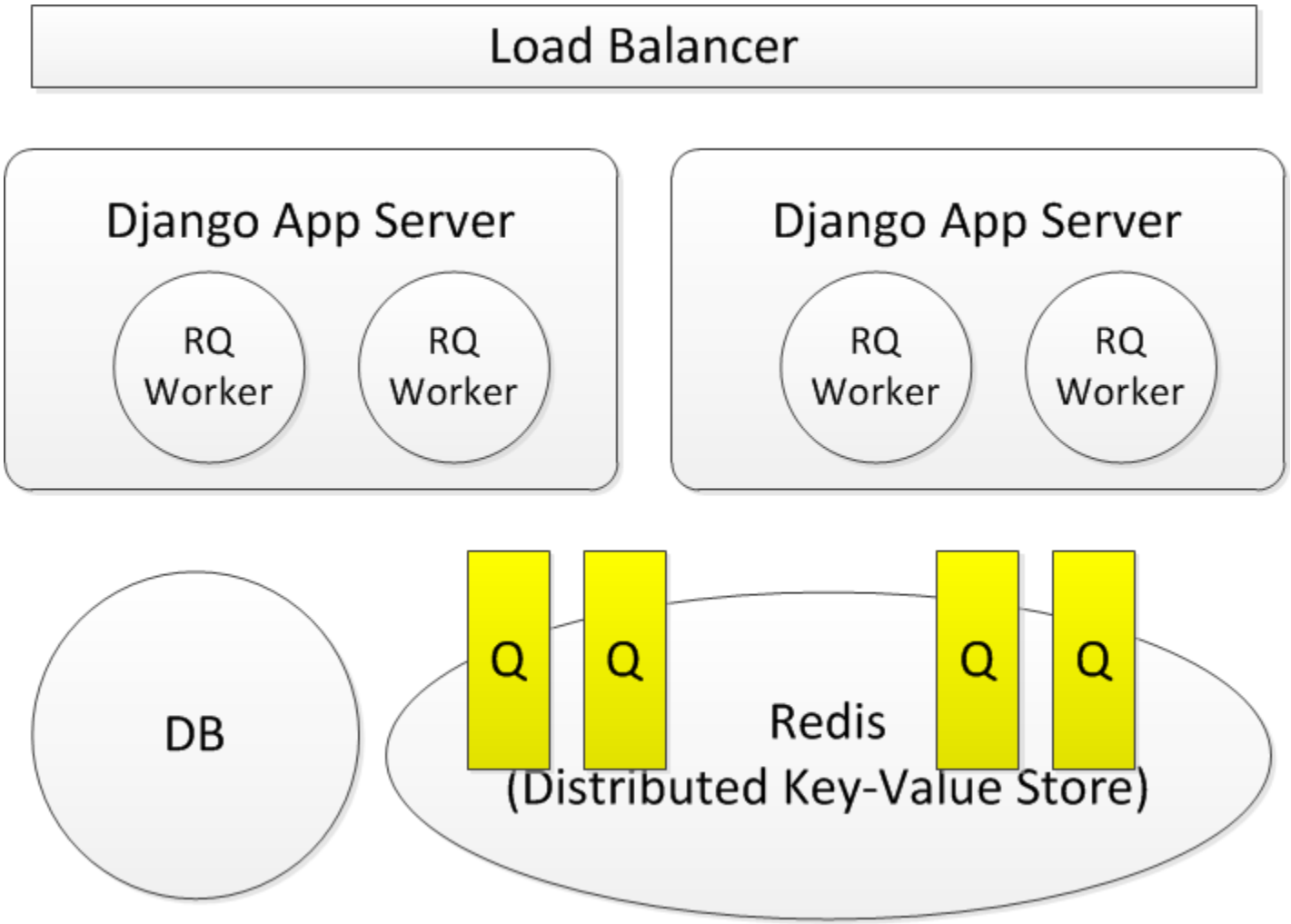
What happens to capacity and user quota values when multiple requests are in flight?



Serialization



Scaling Up with Django-RQ



Before Production

- Safeguards
  - Guard against DDoS runaway scripts: API rate-limiting
  - Test boot-everything-storm resilience
  - Hooking up Django users to LDAP/AD
  - Audit trails – who did what when
- High Availability, independent failure domains
  - Provisioning systems (API servers)
  - Backend systems (e.g NFS HA)
  - not within the cloud artifacts the API server provisions

We are Hiring

Greater Toronto Area

New York

Waterloo

Python, APIs, cloud devops automation, open-stack, kubernetes, cloudify, salt-stack, config-management, frontend UI, Mobile (Android/IoS) and many other areas

jobs.td.com