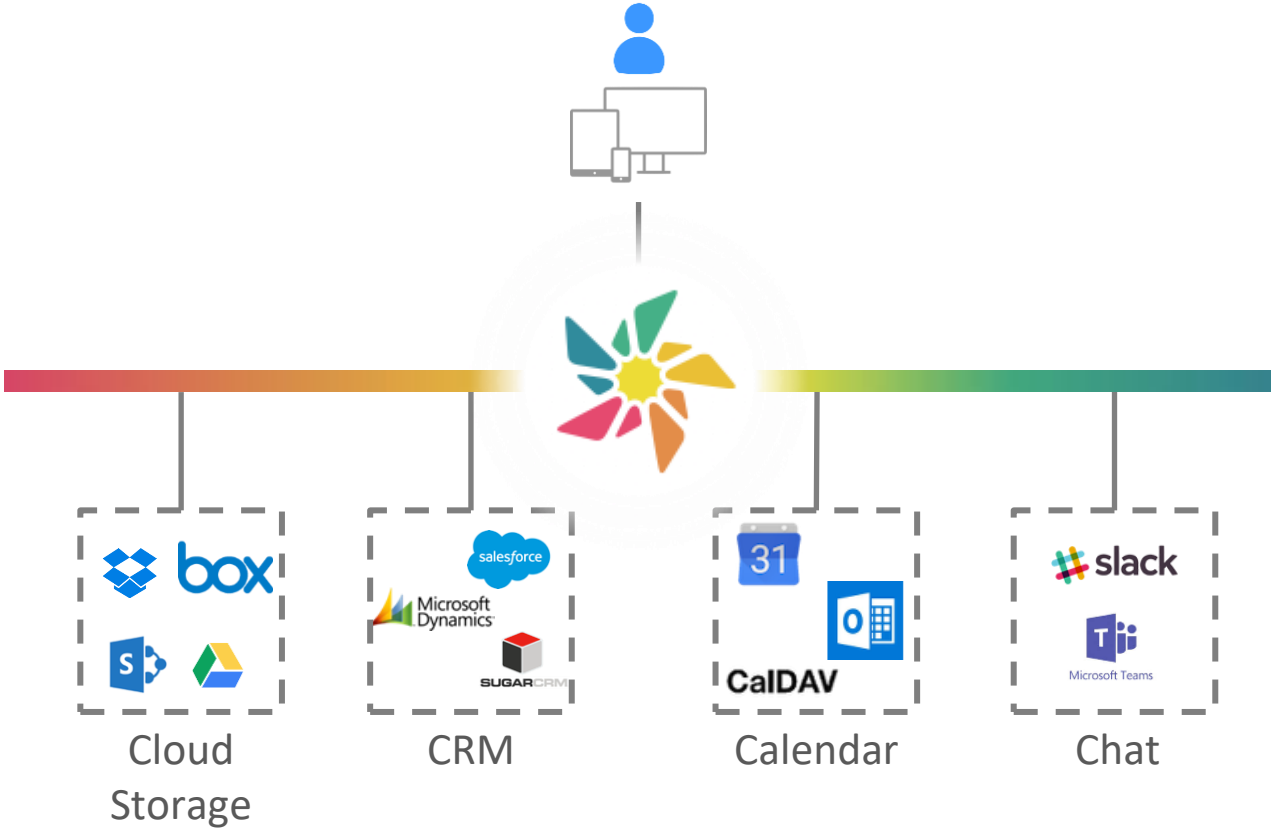
Building APIs with Django REST Framework

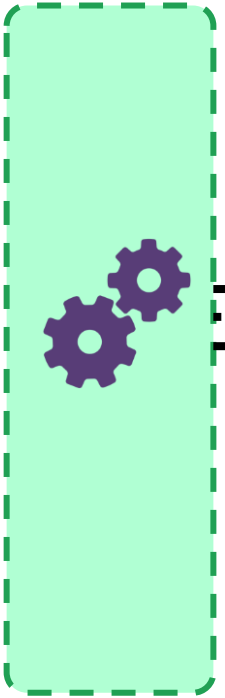
Timothy Liu, VP Engineering
@timothytliu

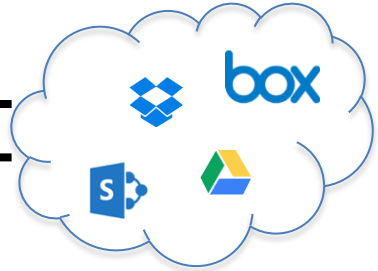

box
salesforce
Microsoft Dynamics
SUGARCRM
31
CalDAV
slack
Microsoft Teams
Cloud Storage
CRM
Calendar
Chat

kloudless

Users

Developer's App



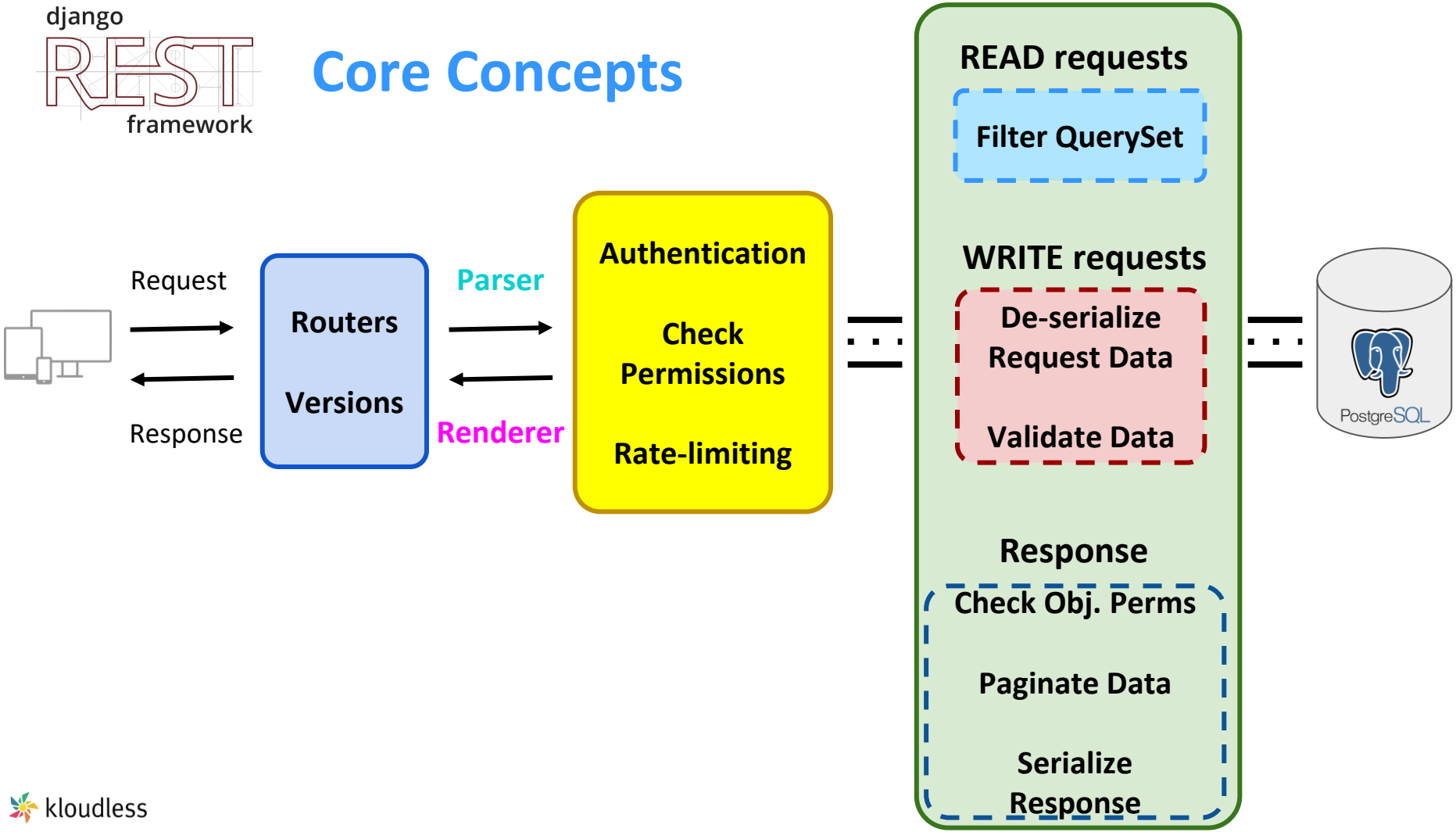


kloudless

Outline

- Core DRF Concepts
- REST: Richardson Maturity Model
  - Level 0: HTTP requests (+JSON response)
  - Level 1: Resources
  - Level 2: HTTP Verbs
- Advanced DRF Concepts
- Questions

Core Concepts



REST: HTTP request and response (w/ JSON)

```
curl -H "Content-Type: application/json"
     -H "Authorization: Bearer TOKEN"
     -X POST -d '{"name": "hello"}'
     "https://api.server.com/v1/accounts"
```

```
{
    "id": 123
}
```

REST: Resources

`"https://api.server.com/v1/accounts"`

Add URLs directly to **`urlpatterns`**.

A Router generates API endpoint URLs based on resources*

```
from rest_framework import routers

router = routers.SimpleRouter()
router.register(r'users', UserViewSet)
router.register(r'accounts', AccountViewSet)
urlpatterns = router.urls
```

kloudless

*URLs in **`urlpatterns`** can also route to DRF views directly, similar to in Django

REST: Resources

"https://api.server.com/v1/accounts"

Each route corresponds to a view method

Each route corresponds to a view method in a **ViewSet** class

```
class AccountViewSet(viewsets.ModelViewSet):
    """
    A simple ViewSet for viewing and editing accounts.
    """
    queryset = Account.objects.all()
    serializer_class = AccountSerializer
```

ViewSets: HTTP Verbs methods ⟺ View

HTTP Verb		ViewSet Method
GET	⟺	.list() .retrieve()
POST	⟺	.create()
PUT	⟺	.update()
PATCH	⟺	.partial_update()
DELETE	⟺	.destroy()

READ requests

ViewSets: Class-based Views

- Routers map to actions. Defaults available for Django models.
- Serializers convert queryset data to/from JSON data.

```
class AccountViewSet(viewsets.ViewSet):
    """A simple ViewSet for listing or retrieving Accounts."""

    def list(self, request):
        queryset = Account.objects.all()
        serializer = AccountSerializer(queryset, many=True)
        return Response(serializer.data)
    def retrieve(self, request, pk=None):
        queryset = Account.objects.all()
        account = get_object_or_404(queryset, pk=pk)
        serializer = AccountSerializer(account)
        return Response(serializer.data)
```


django-rest-framework.org/api-guide/viewsets

READ requests

Filter QuerySet

ViewSets: Using QuerySets

- Query the database.
- Filter the QuerySet returned.

```
from rest_framework import filters, viewsets


class IsOwnerFilterBackend(filters.BaseFilterBackend):
    """Filter that only allows users to see their own objects."""
    def filter_queryset(self, request, queryset, view):
        return queryset.filter(owner=request.user)

class AccountViewSet(viewsets.ModelViewSet):
    """A simple ViewSet for searching owned accounts: GET /accounts?email=kloudless"""
    queryset = Account.objects.all()
    serializer = AccountSerializer

    filter_backends = [filters.SearchFilter, IsOwnerFilterBackend]
    search_fields = ['email']
```

Serializers and Models

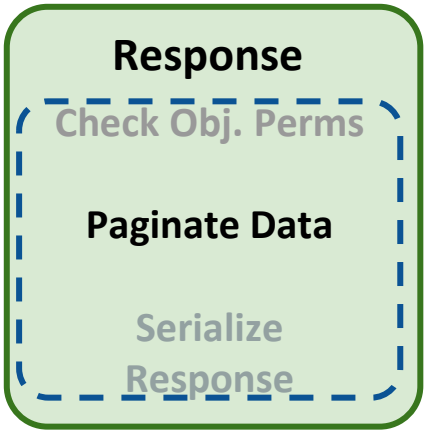
- DRY: Use a Model Serializer
- Validate before model validation

```
from rest_framework import serializers

class AccountSerializer(serializers.ModelSerializer):
    class Meta:
        model = Account
        fields = ['id', 'name', 'custom_properties']

    def validate(self, data):
        if len(data['custom_properties']) > 5:
            raise serializers.ValidationError("custom_properties exceed 5")
        return data

    def create(self, validated_data):
        return Account.objects.create(**validated_data)
```



Pagination

- Use default or define custom pagination
- Serialize response data for list() or ListModelMixin()

```
class CustomPagination(pagination.PageNumberPagination):
    def get_paginated_response(self, data):
        return Response({
            'links': {
                'next': self.get_next_link(),
                'previous': self.get_previous_link()
            },
            'count': self.page.paginator.count,
            'results': data
        })

class AccountViewSet(viewsets.ModelViewSet):
    pagination_class = CustomPagination
```

REST: HTTP request and response (w/ JSON)

Authentication

Check Permissions

Rate-limiting

```
curl -H "Content-Type: application/json"
     -H "Authorization: Bearer TOKEN"
     -X POST -d '{"name": "hello"}'
     "https://api.server.com/v1/accounts"
```

```
429 Too Many Requests
```

Authentication

Check Permissions

Rate-limiting

Authentication: Identifying Users

- Associate request with user
- Handle different authorization schemes

```python
from rest_framework import authentication

class TokenAuthentication(authentication.BaseAuthentication):
    def authenticate(self, request):
        token = authentication.get_authorization_header(request).partition(' ')[-1]

        try:
            user = User.objects.get(token__key=token)
        except User.DoesNotExist:
            raise exceptions.AuthenticationFailed('No such user')

        return (user, token)
```

Authentication

Check Permissions

Rate-limiting

Checking User Permissions

- Check access to this view for any request
- Check access to retrieve, update, or delete a specific object

```python
from rest_framework import permissions

class IsOwnerForModification(permissions.BasePermission):
    """Only allow write access to the object's owner"""
    def has_object_permission(self, request, view, obj):
        # GET, HEAD, OPTIONS allowed
        if request.method in permissions.SAFE_METHODS:
            return True

        return obj.owner == request.user
```


django-rest-framework.org/api-guide/permissions

Authentication

Check Permissions

Rate-limiting

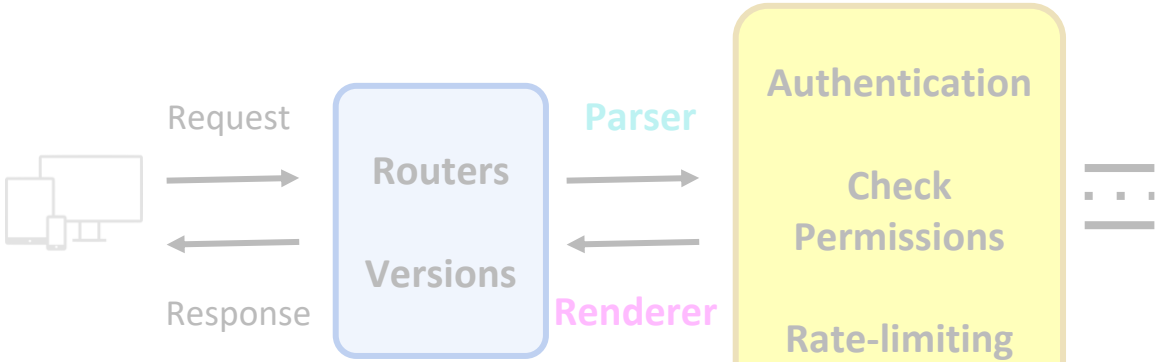
Rate-limiting User Requests

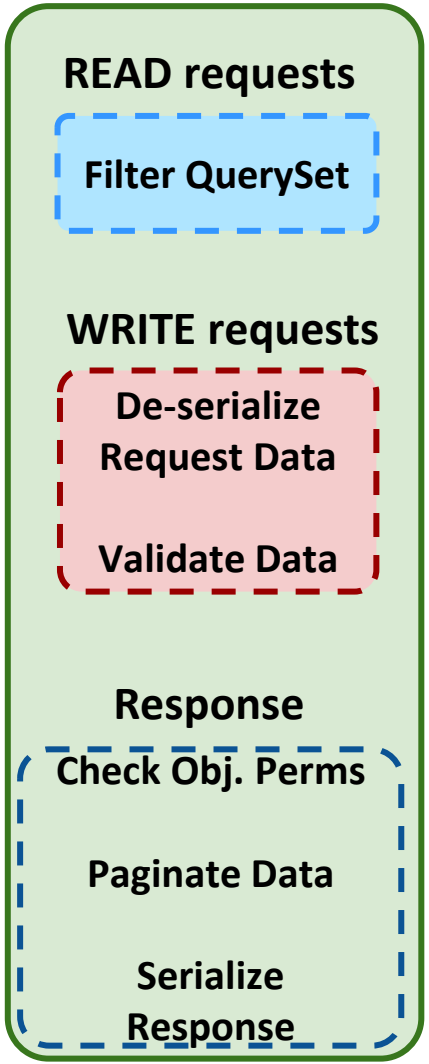
- Per-user throttles: **Burst** (e.g. 10/s), **Sustained** (e.g. 10,000/day)
- Can be scoped per View, or include custom throttling like below.

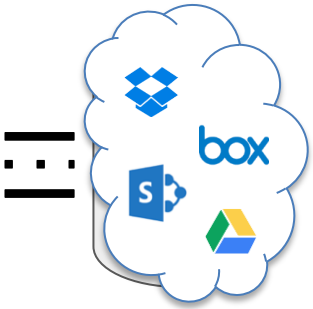
```
from rest_framework import throttling

class CosmicRayRateThrottle(throttling.BaseThrottle):
    """Simulate a stray cosmic ray flipping a bit that rate-limits this request"""
    def allow_request(self, request, view):
        return random.randint(1, 10e12) == 1
```

Advanced Concepts







Custom Endpoints and QuerySets

- DRF allows custom routes and view methods
- Return custom JSON response based on the upstream service

```
class KloudlessFileQuerySet(object):
    def copy(self, *args, **kwargs):
        ... # copy file in cloud service

class CustomFileViewSet(viewsets.ViewSet):
    queryset = KloudlessFileQuerySet()

    @action(detail=True, methods=['post'])
    def copy(self, request):
        queryset = self.filter_queryset(self.get_queryset())

        obj = queryset.copy(self.lookup, **params)

        return response.Response(self.get_serializer(obj).data, status=201)
```

Thanks

@timothytliu

kloudless