DEV DAY
TOKYO

B - 4

# Deep Dive on AWS SDK for Python (Boto3)

Kyle Knapp (@thekyleknapp)
Software Development Engineer
AWS/AWS SDKs and Tools

2019.10.03-04

aws

# Agenda

- Introduction to Boto3
- Dynamic client generation
- Event system
- Sessions
- Conclusion

DEV DAY

# Introduction to Boto3

# Boto3 usage

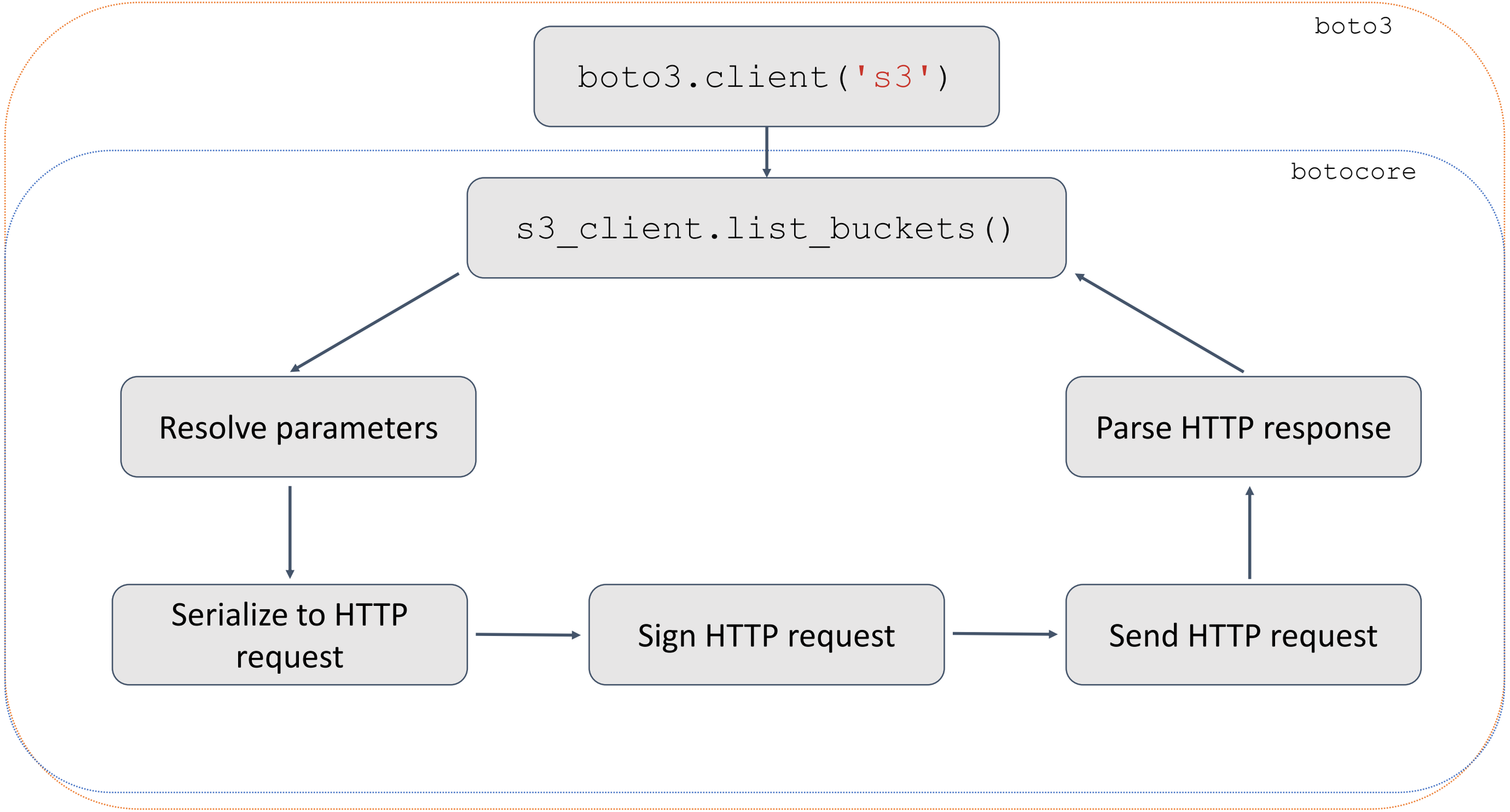
```
$ pip install boto3
```

# Boto3 usage

```
import boto3

s3_client = boto3.client('s3')
print(s3_client.list_buckets())
```

# Boto3 usage

```
{
 'Buckets': [{
  'CreationDate': datetime.datetime(2016, 4, 24, 22, 36, 20)
   Name': 'my-bucket'
  }],
 'Owner’: {
  'DisplayName': 'user',
  'ID': '876e6c081ece589308db3ea34172…'},
 'ResponseMetadata': {‘<Content shortened>'}
}
```


boto3
boto3.client('s3')
botocore
s3_client.list_buckets()
Resolve parameters
Serialize to HTTP request
Sign HTTP request
Send HTTP request
Parse HTTP response

# Agenda

- Introduction to Boto3
  - ✓ `pip install boto3`
  - ✓ Client usage
  - ✓ `boto3` is a small wrapper around `botocore`
- Dynamic client generation
- Event system
- Sessions
- Conclusion

DEV DAY

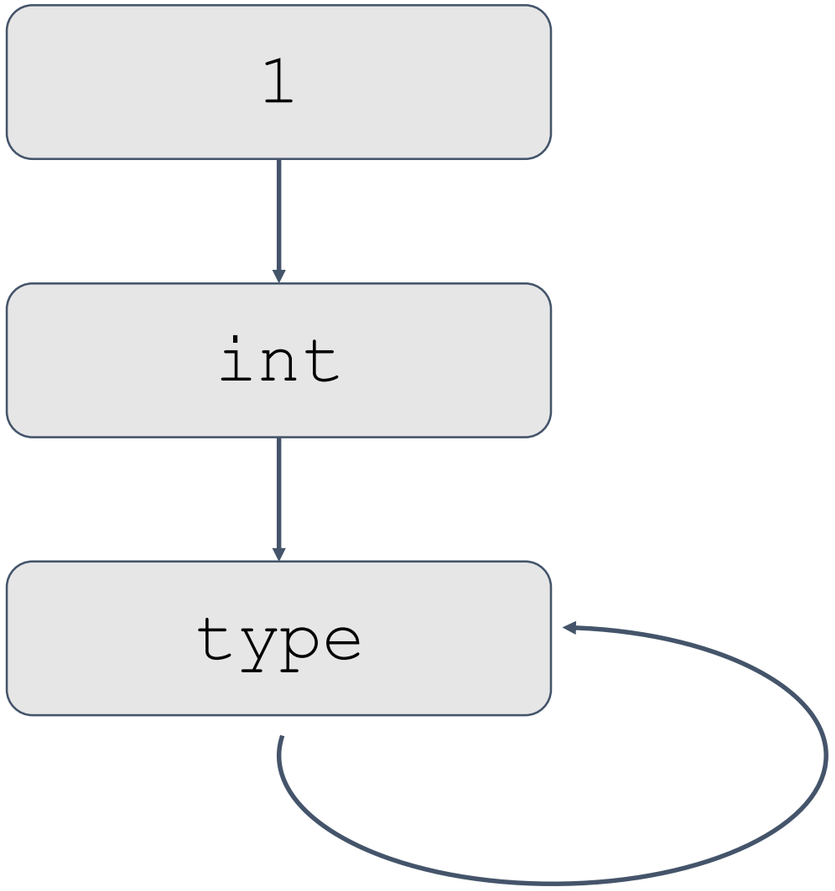
# Dynamic client generation

# Dynamic class generation

```
$ git clone https://github.com/boto/botocore.git
$ grep -r 'def list_buckets' botocore
```

# Dynamic class generation: `type()`

```
>>> type(1)
<class 'int'>

>>> type(int)
<class 'type'>

>>> type(type)
<class 'type'>
```



# Dynamic class generation: `type()`

```
type(name, bases, properties)
```

Name of the class

Tuple of base classes

Dictionary of properties

# Dynamic class generation: `type()`

```
def hello_world(obj):
    print('Hello World from %s!' % obj)

class BaseClass:
    def __init__(self, name):
        self.name = name

class MyClass(BaseClass):
    hello_world = hello_world
```

```
def hello_world(obj):
    print('Hello World from %s!' % obj)

class BaseClass:
    def __init__(self, name):
        self.name = name

MyClass = type(
    'MyClass',
    (BaseClass,),
    {'hello_world': hello_world}
)
```

```
>>> my_instance = MyClass('My Instance')
>>> print(my_instance.name)
My Instance
>>> my_instance.hello_world()
Hello World from My Instance!
```

# Dynamic class generation: Benefits

- Reduces physical size
- Improves development efficiency of library
- Improves reliability of library

# Dynamic class generation: botocore

```
$ tree -L 1 botocore/botocore/data
botocore/botocore/data/
|── acm
|── acm-pca
|── alexaforbusiness
|── amplify
|── apigateway
...
|── s3
...
|── xray

191 directories, 2 files
```

```
boto3.client('s3')
```

# Dynamic class generation: `botocore`

```
$ tree botocore/botocore/data/s3
botocore/botocore/data/s3
|── 2006-03-01
    |── examples-1.json
    |── paginators-1.json
    |── service-2.json
    |── waiters-2.json


1 directory, 4 files
```

```
boto3.client('s3')
```

# Dynamic class generation: botocore

```
s3 = boto3.client('s3')
```

```
s3.abort_multipart_upload()
s3.complete_multipart_upload()
s3.copy_object()
s3.create_bucket()

s3.list_buckets()
```

```
{
  "version":"2.0",
  "metadata":{...},
  "operations":{
    "AbortMultipartUpload":{...},
    "CompleteMultipartUpload":{...},
    "CopyObject":{...},
    "CreateBucket":{...},
    ...
    "ListBuckets":{...},
    ...
  }
}
```

botocore/data/s3/2006-03-01/service-2.json

# Dynamic class generation: botocore

```
response = s3.list_buckets()
```

```
{
  "version":"2.0",
  "metadata":{...},
  "operations":{
    ...
    "ListBuckets":{
      "name":"ListBuckets",
      "http":{
        "method":"GET",
        "requestUri":"/"
      },
      "output":{"shape":"ListBucketsOutput"}
  }
}
```

botocore/data/s3/2006-03-01/service-2.json

# Dynamic class generation: botocore

```
>>> print(s3.list_buckets())
{
  'Buckets': [{…}],
  'Owner': {…},
}
```

```
{
  "shapes":{
    "ListBucketsOutput":{
      "type":"structure",
      "members":{
        "Buckets":{
          "shape":"Buckets",
        },
        "Owner":{
          "shape":"Owner",
        }
      }
    }
  }
}
```

botocore/data/s3/2006-03-01/service-2.json

# Dynamic class generation: botocore

```
class ClientCreator(object):
    def create_client_class(self, service_name, api_version=None):
        service_model = self._load_service_model(service_name, api_version)
        return self._create_client_class(service_name, service_model)

    def _create_client_class(self, service_name, service_model):
        class_attributes = self._create_methods(service_model)
        py_name_to_operation_name = self._create_name_mapping(service_model)
        class_attributes['_PY_TO_OP_NAME'] = py_name_to_operation_name
        bases = [BaseClient]
        service_id = service_model.service_id.hyphenize()
        self._event_emitter.emit(
            'creating-client-class.%s' % service_id,
            class_attributes=class_attributes,
            base_classes=bases)
        class_name = get_service_module_name(service_model)
        cls = type(str(class_name), tuple(bases), class_attributes)
        return cls
```

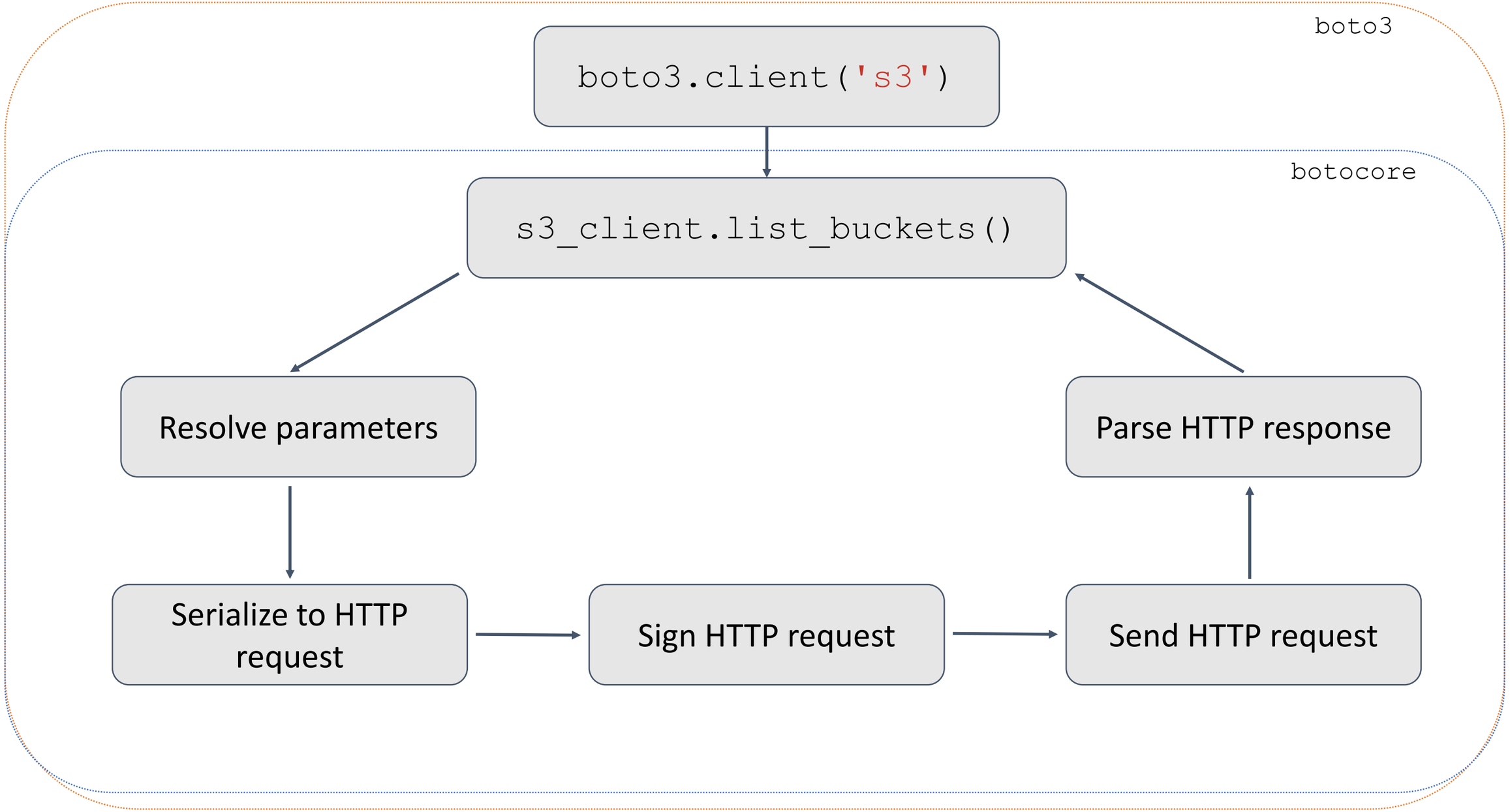
botocore/client.py

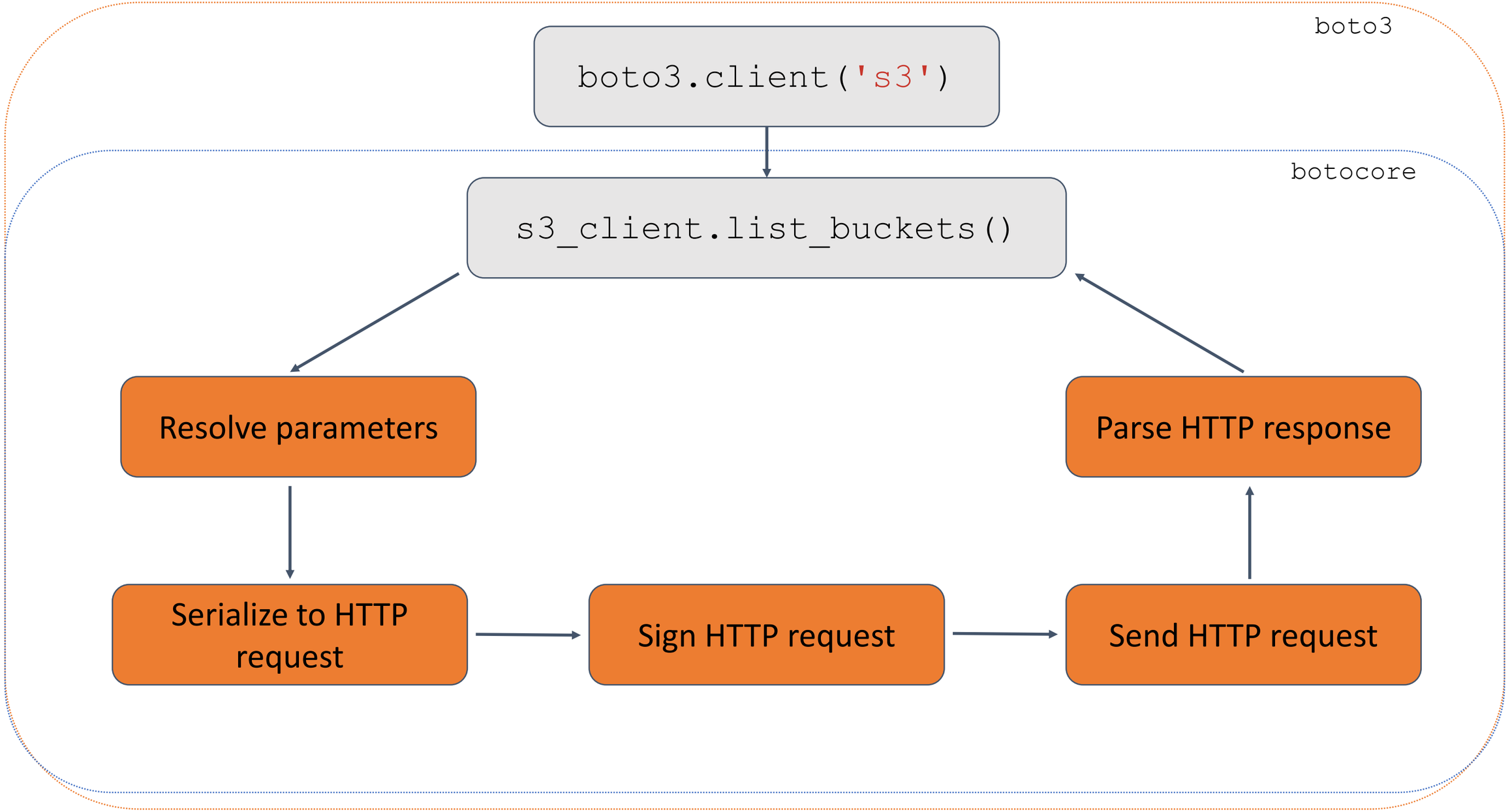
# Agenda

- Introduction to Boto3
- Dynamic client generation
  - ✓ `botocore` dynamically creates classes from JSON models
  - ✓ Benefits of dynamic client generation
- Event system
- Sessions
- Conclusion

DEV DAY

# Event system


boto3
boto3.client('s3')
botocore
s3_client.list_buckets()
Resolve parameters
Serialize to HTTP request
Sign HTTP request
Send HTTP request
Parse HTTP response


boto3
boto3.client('s3')
botocore
s3_client.list_buckets()
Resolve parameters
Serialize to HTTP request
Sign HTTP request
Send HTTP request
Parse HTTP response

# Event system: Sample use case

```
import boto3

client = boto3.client('lambda')
print(client.get_policy(FunctionName='myapp-dev'))

{'ResponseMetadata': {…},
 'Policy': '{"Version":"2012-10-
17","Id":"default","Statement":[{"Sid":"96ae78f3-4e4b-4f7d-b0f2-
b3b3005f0842","Effect":"Allow","Principal":{"Service":"apigateway.amazonaws.
com"},"Action":"lambda:InvokeFunction","Resource":"arn:aws:lambda:us-west-
2:123456789123:function:myapp-
dev","Condition":{"ArnLike":{"AWS:SourceArn":"arn:aws:execute-api:us-west-
2:123456789123:snfv76gjii/*"}}}]}', 'RevisionId': '98c521e9-0f20-4401-af55-
23ad133aa3e5'
}
```

# Event system: Sample use case

```
{
  "version":"2.0",
  "metadata":{...},
  "shapes":{
    "GetPolicyResponse":{
      "type":"structure",
      "members":{
        "Policy":{
          "shape":"String",
        },
        "RevisionId":{
          "shape":"String",
        }
      }
    }
  }
}
```

botocore/data/lambda/2015-03-31/service-2.json

# Event system: Sample use case

```
import json
import boto3

client = boto3.client('lambda')
response = client.get_policy(FunctionName='myapp-dev')
response['Policy'] = json.loads(response['Policy'])
print(response)
```

```
{'ResponseMetadata': {…},
 'Policy': {'Version': '2012-10-17', 'Id': 'default', 'Statement': [{'Sid':
'96ae78f3-4e4b-4f7d-b0f2-b3b3005f0842', 'Effect': 'Allow', 'Principal':
{'Service': 'apigateway.amazonaws.com'}, 'Action': 'lambda:InvokeFunction',
'Resource': 'arn:aws:lambda:us-west-2:123456789123:function:myapp-dev',
'Condition': {'ArnLike': {'AWS:SourceArn': 'arn:aws:execute-api:us-west-
2:123456789123:snfv76gjii/*'}}}]}, 'RevisionId': '98c521e9-0f20-4401-af55-
23ad133aa3e5'
}
```

# Event system: Sample use case

```
s3_client.meta.events.register(event_name, handler)
```

# Event system: Sample use case

```
s3_client.meta.events.register(event_name, handler)
```

**Pattern:** `<event-type>.<service-name>.<operation-name>`

**Example values:**

- `after-call`
- `after-call.lambda`
- `after-call.lambda.GetPolicy`

# Event system: Sample use case

```
s3_client.meta.events.register(event_name, handler)
```

```
def handler(**kwargs):
    print('Called handler with %s' % kwargs)
```

# Event system: Sample use case

```
import json
import boto3

def load_policy(parsed, **kwargs):
    parsed['Policy'] = json.loads(parsed['Policy'])

client = boto3.client('lambda')
client.meta.events.register(
    'after-call.lambda.GetPolicy', load_policy)
print(client.get_policy(FunctionName='myapp-dev'))
```

```
{'ResponseMetadata': {…},
 'Policy': {'Version': '2012-10-17', 'Id': 'default', 'Statement': [{'Sid':
'96ae78f3-4e4b-4f7d-b0f2-b3b3005f0842', 'Effect': 'Allow', 'Principal':
{'Service': 'apigateway.amazonaws.com'}, 'Action': 'lambda:InvokeFunction',
'Resource': 'arn:aws:lambda:us-west-2:123456789123:function:myapp-dev',
'Condition': {'ArnLike': {'AWS:SourceArn': 'arn:aws:execute-api:us-west-
2:123456789123:snfv76gjii/*'}}}]}, 'RevisionId': '98c521e9-0f20-4401-af55-
23ad133aa3e5'
}
```

# Event system: Available events

```
import boto3


lambda_client = boto3.client('lambda')


def print_event(event_name, **kwargs):
    print(event_name)


lambda_client.meta.events.register('*', print_event)
lambda_client.get_policy(FunctionName='myapp-dev')
```

# Event system: Available events

```
provide-client-params.lambda.GetPolicy
before-parameter-build.lambda.GetPolicy
before-call.lambda.GetPolicy
request-created.lambda.GetPolicy
choose-signer.lambda.GetPolicy
before-sign.lambda.GetPolicy
before-send.lambda.GetPolicy
response-received.lambda.GetPolicy
needs-retry.lambda.GetPolicy
after-call.lambda.GetPolicy
```

# Event system: Available events

```
provide-client-params.lambda.GetPolicy
before-parameter-build.lambda.GetPolicy
before-call.lambda.GetPolicy
request-created.lambda.GetPolicy
choose-signer.lambda.GetPolicy
before-sign.lambda.GetPolicy
before-send.lambda.GetPolicy
response-received.lambda.GetPolicy
needs-retry.lambda.GetPolicy
after-call.lambda.GetPolicy
```

# Agenda

- Introduction to Boto3
- Dynamic client generation
- Event system
  - ✓ Events allows for the modification of lifecycle for API request
  - ✓ Can register handlers on a event, service, operation granularity
- Sessions
- Conclusion

DEV DAY

# Sessions

# Sessions: Introduction

```python
import boto3


s3_client = boto3.client('s3')
print(s3_client.list_buckets())
```

# Sessions: Introduction

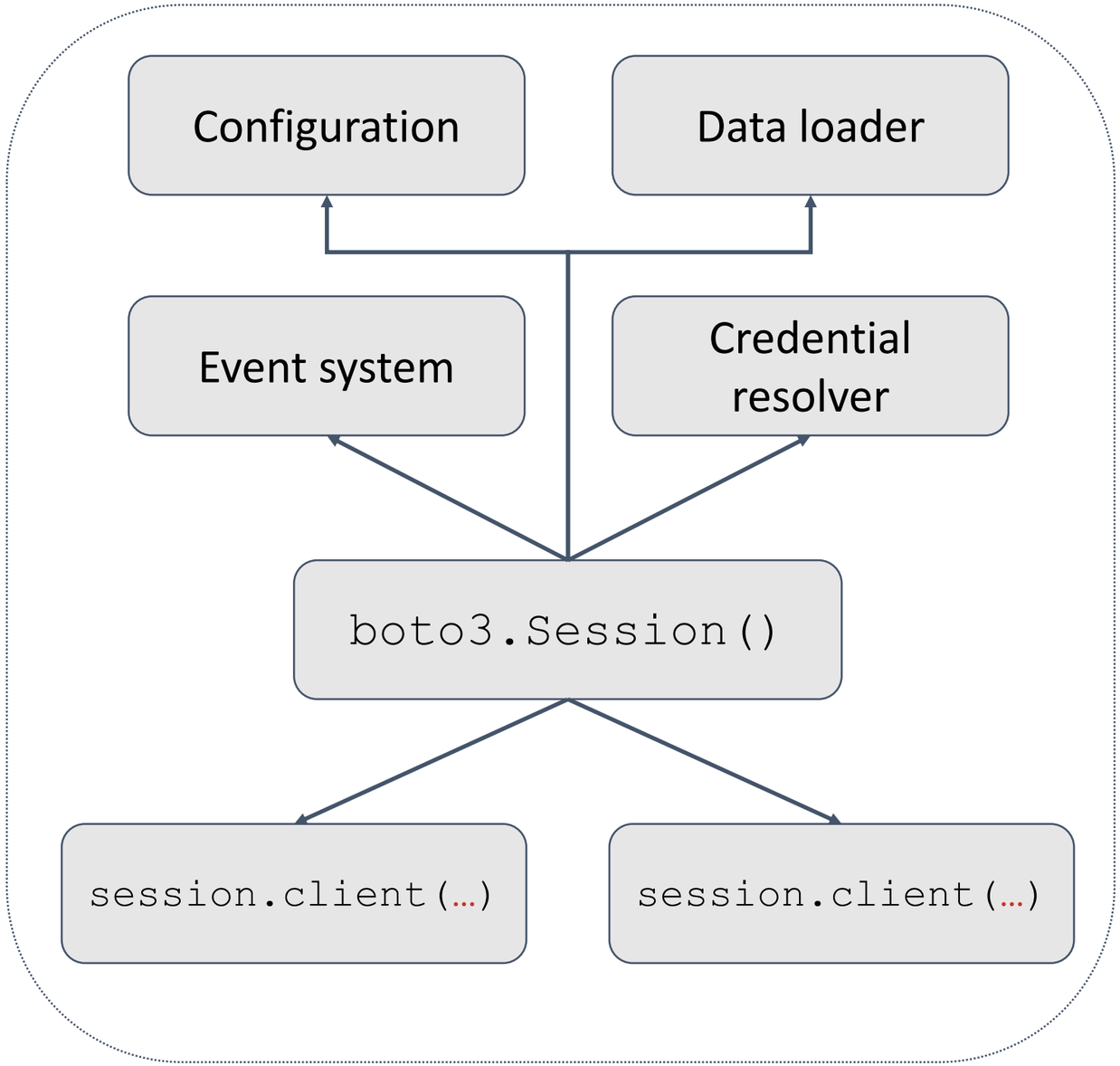
```
import boto3


session = boto3.Session()
s3_client = session.client('s3')
print(s3_client.list_buckets())
```

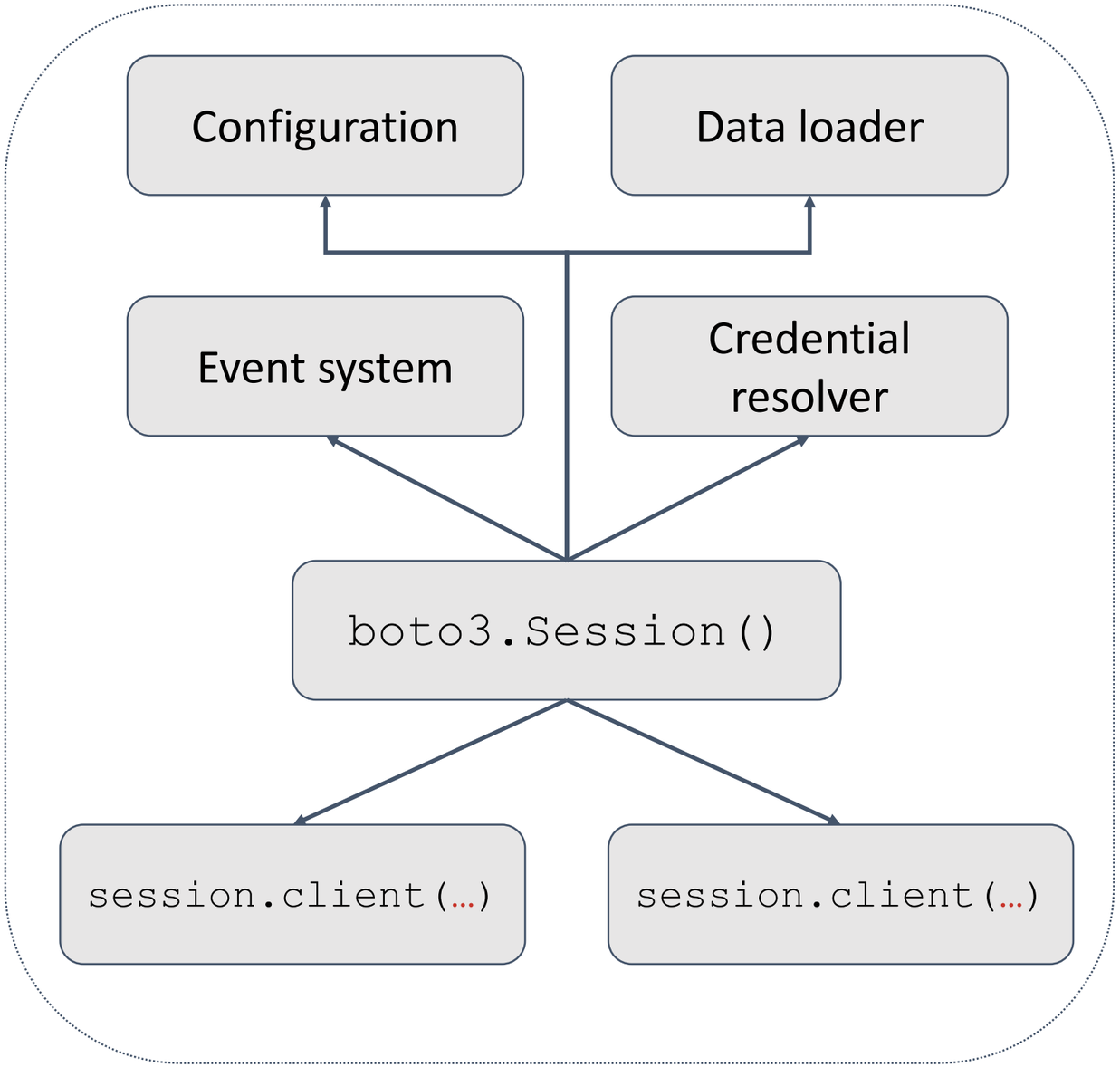
# Sessions: Introduction

```python
import boto3

s3_client = boto3.client('s3')
```

```python
def client(*args, **kwargs):
    return _get_default_session().client(
        *args, **kwargs)


def _get_default_session():
    if DEFAULT_SESSION is None:
        setup_default_session()

    return DEFAULT_SESSION
```

boto3/__init__.py


Configuration
Data loader
Event system
Credential resolver
boto3.Session()
session.client(…)
session.client(…)


Configuration
Data loader
Event system
Credential resolver
boto3.Session()
session.client(…)
session.client(…)

# Sessions: Usage recommendations

- Use `Session` objects to unify behavior across clients
- Limit the number of `Session` objects
- Use `botocore.session.Session` directly to affect lower level functionality.

# Session component: Credential resolver

```
resolver = CredentialResolver([
    EnvProvider(...),
    AssumeRoleProvider(...),
    AssumeRoleWithWebIdentityProvider(...),
    SharedCredentialProvider(...),
    ProcessProvider(...),
    ConfigProvider(...),
    ContainerProvider(...),
    InstanceMetadataProvider(...)
])
```

botocore/credentials.py

```
resolver.load_credentials()
```

```
load()
load()
load()
load()
```

Credentials

# Session: InstanceMetadataProvider

```
class InstanceMetadataProvider(CredentialProvider):
    METHOD = 'iam-role'
    CANONICAL_NAME = 'Ec2InstanceMetadata'

    def __init__(self, iam_role_fetcher):
        self._role_fetcher = iam_role_fetcher

    def load(self):
        metadata = self._role_fetcher.retrieve_iam_role_credentials()
        if not metadata:
            return None
        creds = RefreshableCredentials.create_from_metadata(
            metadata,
            method=self.METHOD,
            refresh_using=fetcher.retrieve_iam_role_credentials,
        )
        return creds
```

botocore/credentials.py

# Session: Adding a custom provider

```
from botocore.session import Session

session = Session()
creds_resolver = session.get_component('credential_provider’)
custom_provider = MyCustomProvider()
creds_resolver.insert_after('env', custom_provider)

s3_client = session.create_client('s3')
```

# Sessions: Other useful functionality

- Event system: `Session.get_component('event_emitter')`
- Client config: `Session.set_default_client_config(Config())`
- Region enumeration: `Session.get_available_regions('<service-name>')`

# Agenda

- Introduction to Boto3
- Dynamic client generation
- Event system
- Sessions
  - ✓ Affects functionality across a group of clients
  - ✓ Credential resolver
  - ✓ Other useful functionality
- Conclusion

# Conclusion

# Conclusion: Topics covered

- Boto3/Botocore usage
- Dynamic client generation
- Event system
- Sessions

# Thank you!

- GitHub repositories:
  - Boto3: https://github.com/boto/boto3
  - Botocore: https://github.com/boto/botocore

- Documentation:
  - Boto3: https://boto3.amazonaws.com/v1/documentation/api/latest/index.html
  - Botocore: https://botocore.amazonaws.com/v1/documentation/api/latest/index.html

Kyle Knapp (@thekyleknap)
https://github.com/kyleknap/