

Harness analytical strengths: SAS, R, Python Web Services

Trends and Technology – TT11

Sandeep Juneja, Ben Bocchicchio

SAS Institute, Cary, North Carolina, USA



Harness analytical strengths: SAS, R, Python Web Services

Biography



Ben is an Advisory Technical Consultant at SAS and has more than 25 year of experience in designing and implementing solutions for various Health Care and Life Science clients.



Sandeep is AWS & Azure certified Solutions Architect. He is Senior Consulting Manager at SAS and has more than 17 years of experience in designing and implementing Cloud based solutions for various Health Care and Life Science clients.

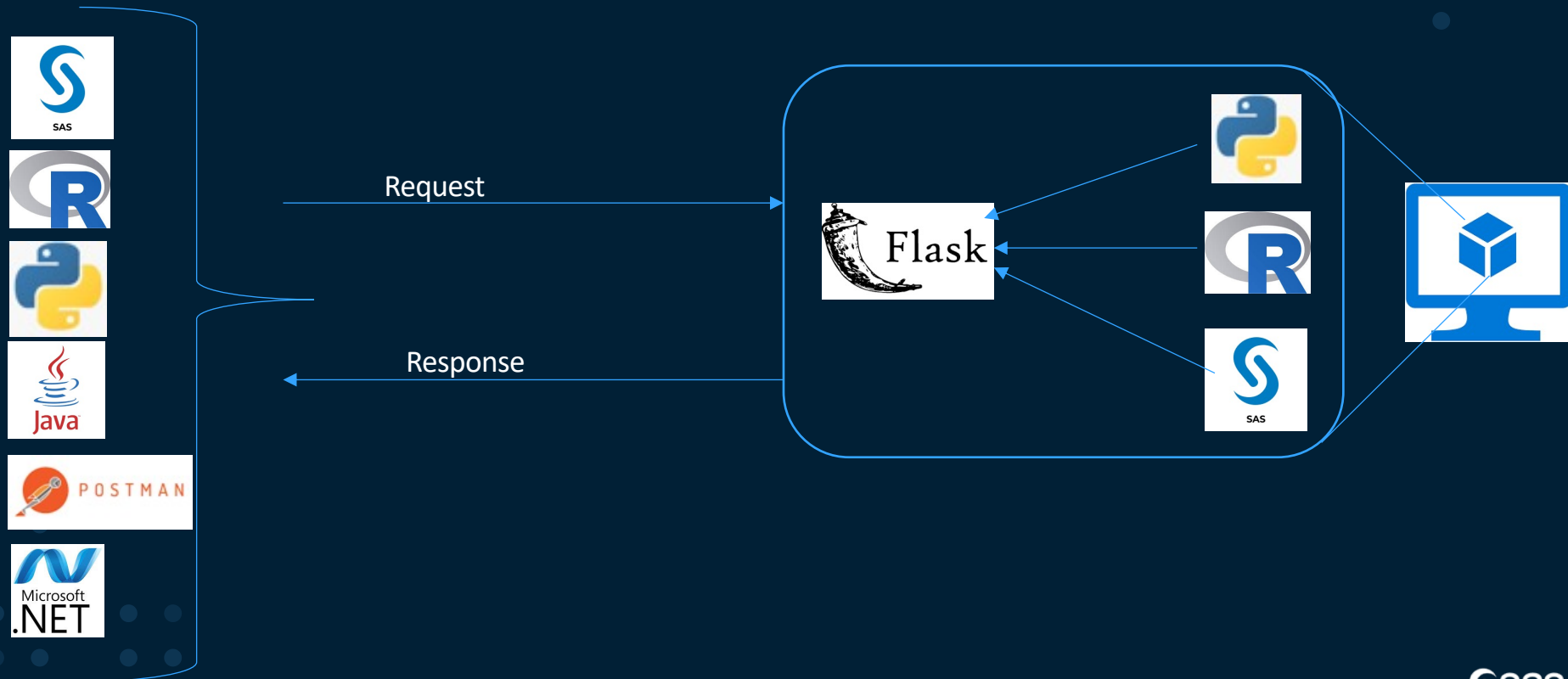
Harness analytical strengths: SAS, R, Python Web Services

Use Cases

	Description	Python	R	SAS
1	Work in cloud environment. Don't have access to MS office and need to convert Word files to pdf	✓		
2	Work in qualified environment and want to test some R code with new packages		✓	
3	Have output in Word file and need to convert it to SAS dataset for validation purpose	✓		✓

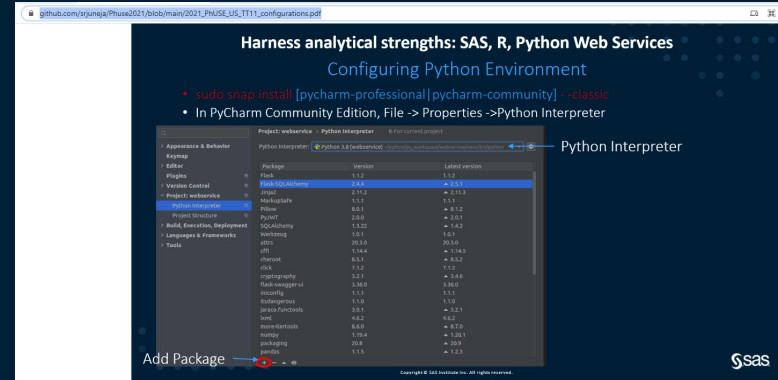
Harness analytical strengths: SAS, R, Python Web Services

Architecture



Configuring Virtual Machine

- Configuration Slides for setting up
 - Virtual Machine
 - Python
 - R
 - SAS
 - SASPy



Harness analytical strengths: SAS, R, Python Web Services

Webservice project – Why Flask / Blueprint?

- In Flask, views can be any callable (like a function) that receives requests and returns the response for that request. Flask is responsible for sending the response back to the user.
- Flask Blueprints encapsulate functionality, such as views, templates, and other resources.
- Similar to API Management

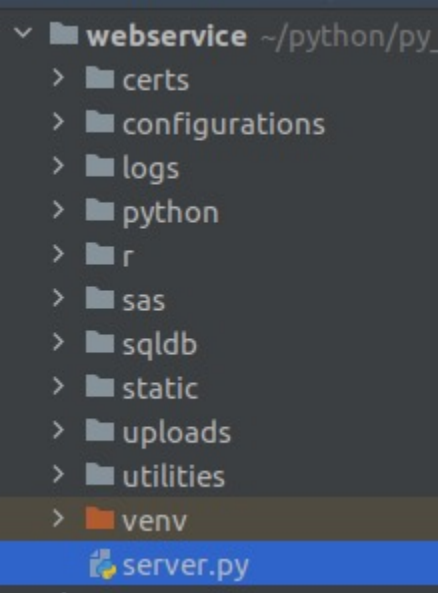
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
    return "Webservice app"
```

```
from flask import Flask
from sas.sas_main import sas
app = Flask(__name__)
app.register_blueprint(sas)

@sas.route('/sas', methods=['GET'])
def hello():
    return {"data": "Hello SAS Webservices!"}
```

Harness analytical strengths: SAS, R, Python Web Services

Project Structure



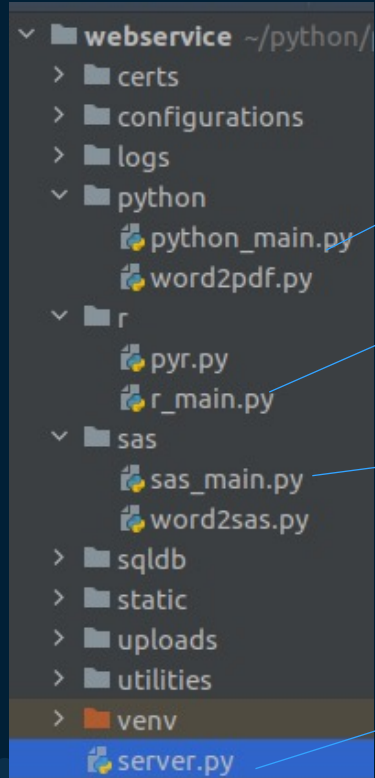
- server.py – Pyflask server
- python – register Python web services
- r – register R web services
- sas - register SAS web services
- configurations – configuration options
- venv – virtual environment
- logs – capture incoming request details in logs
- static – swagger docs
- sqldb – user credentials in sql lite db
- certs – support HTTPS
- uploads – intermediate files uploaded by client requests

Source Code : <https://github.com/srjuneja/Phuse2021>

Harness analytical strengths: SAS, R, Python Web Services

Webservice project – Register SAS, R, Python Blueprints

- Define Python, R and SAS webservice files and register blueprints in server.py



```
python = Blueprint("python", __name__, root_path="/python")

@python.route('/python', methods=['GET'])
def hello():
    return {"data": "Hello Python Webservices!"}
```

```
r = Blueprint("r", __name__, root_path="/r")

@r.route('/r', methods=['GET'])
def hello():
    return {"data": "Hello R Webservices!"}
```

```
sas = Blueprint("sas", __name__, root_path="/sas")

@sas.route('/sas', methods=['GET'])
def hello():
    return {"data": "Hello SAS Webservices!"}
```

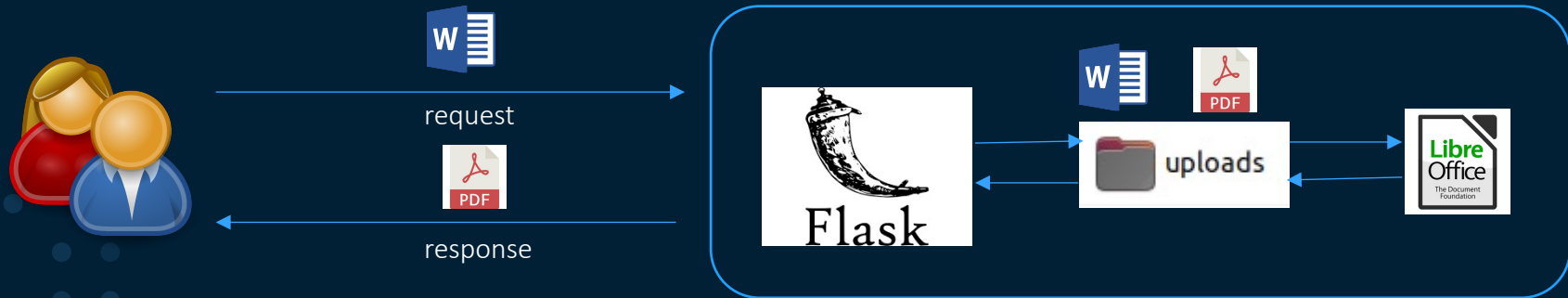
```
from python.python_main import python
from r.r_main import r
from sas.sas_main import sas

# Register BluePrints
app.register_blueprint(python)
app.register_blueprint(r)
app.register_blueprint(sas)
```

Harness analytical strengths: SAS, R, Python Web Services

Use Case #1 - Python webservice - word2pdf

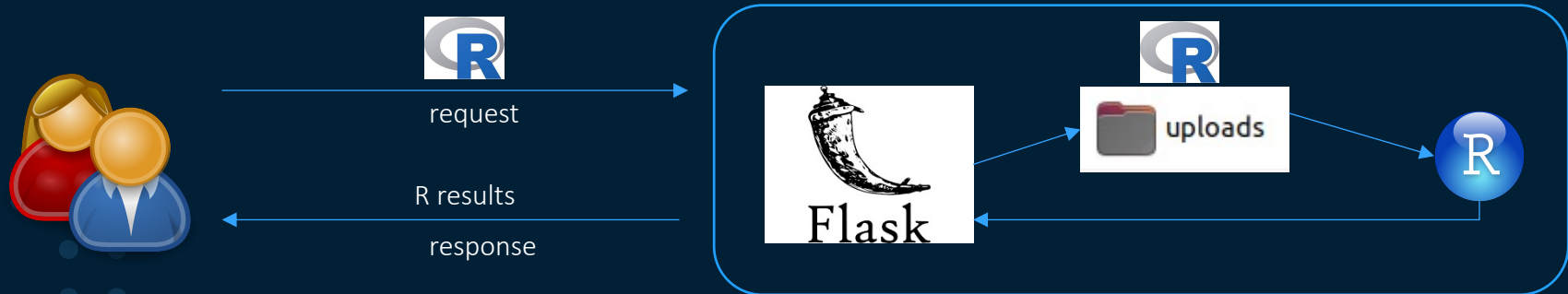
- Convert Word file to pdf file
 - User sends Word file in Request
 - File gets uploaded to uploads folder
 - Using Libre Office word file is converted to pdf file
 - Pdf file is sent as Response back to User
 - Both word and pdf files are deleted from uploads folder



Harness analytical strengths: SAS, R, Python Web Services

Use Case #2 - R Webservice - Run r_code

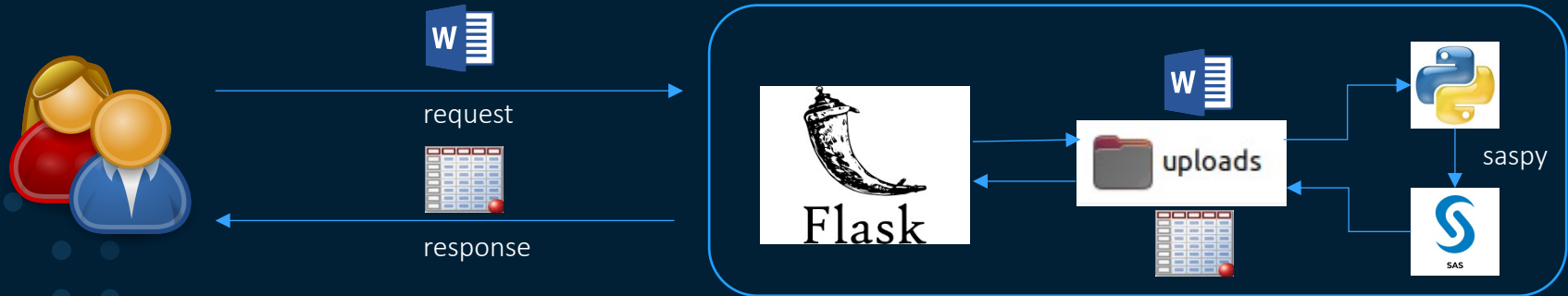
- Execute R program and capture results
 - User sends R program in Request
 - File gets uploaded to uploads folder
 - Using SubProcess concept, R program is submitted to R language
 - Results of R execution is streamed back in Response
 - R program is deleted from uploads folder



Harness analytical strengths: SAS, R, Python Web Services

Use Case #3 - SAS Webservice - word2sas

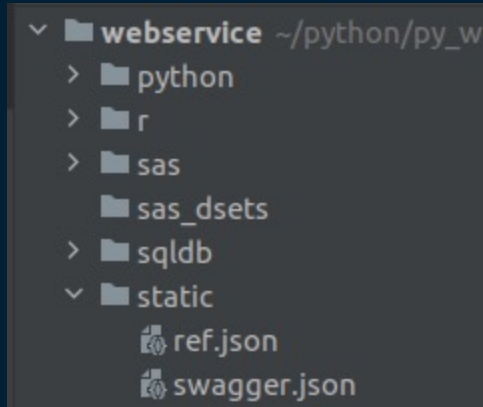
- Convert word table to sas dataset
 - User sends Word file in Request
 - File gets uploaded to uploads folder
 - Using Python docx package read file into DataFrames
 - Using saspy initialize connection to SAS and convert DF to sas dataset
 - SAS dataset is sent as Response back to User
 - SAS session is killed
 - Word file and sas dataset are deleted from uploads folder



Harness analytical strengths: SAS, R, Python Web Services

Swagger Docs

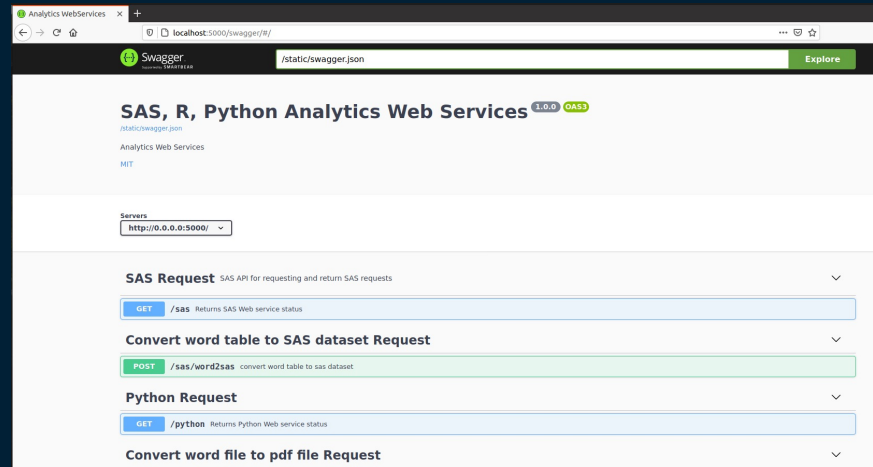
- Documenting Endpoints



```
# Documentation - Swagger
SWAGGER_URL = '/swagger'
API_URL = '/static/swagger.json'
```

```
SWAGGERUI_BLUEPRINT = get_swaggerui_blueprint(
    config.SWAGGER_URL,
    config.API_URL,
    config={
        'app_name': "Analytics WebServices"
    }
)

app.register_blueprint(SWAGGERUI_BLUEPRINT, url_prefix=config.SWAGGER_URL)
```



Harness analytical strengths: SAS, R, Python Web Services

Multi-Threading

- Flask is single threaded.
- WSGIserver is a high-speed, production ready, thread pooled, generic WSGI server with SSL support.
 - <https://pypi.org/project/WSGIserver/>
 - <https://docs.cherrypy.org/en/3.2.6/refman/wsgiserver/init.html#>
 - <https://cheroot.cherrypy.org/en/latest/index.html#>

```
if __name__ == '__main__':  
    |  
    server = WSGIServer(bind_addr=("0.0.0.0", int(5000)), wsgi_app=app, numthreads=100)  
    try:  
        print("Starting Server at :" + strftime("%H:%M:%S"))  
        server.start()  
    except KeyboardInterrupt:  
        server.stop()
```

Harness analytical strengths: SAS, R, Python Web Services

Securing Endpoints - Authentication

- Implement SQL Lite DB to store user credentials and implement authentication requirements before accessing endpoints

<https://geekflare.com/securing-flask-api-with-jwt/>

```
# DATABASE PARAMETERS
SQLALCHEMY_DATABASE_URI="sqlite:///{}../sqldb/data.db".format(PROJECT_HOME)
SQLALCHEMY_TRACK_MODIFICATIONS = True
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = config.SQLALCHEMY_DATABASE_URI
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = True
db = SQLAlchemy(app)
```

```
class Users(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    public_id = db.Column(db.Integer)
    name = db.Column(db.String(50))
    password = db.Column(db.String(50))
    admin = db.Column(db.Boolean)
```

Harness analytical strengths: SAS, R, Python Web Services

Securing Endpoints - Authentication

- Define token decorator and apply to secure route
- <https://geekflare.com/securing-flask-api-with-jwt/>

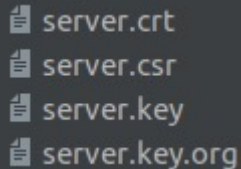
```
def token_required(f):  
    @wraps(f)  
    def decorator(*args, **kwargs):  
        token = None  
        if 'x-access-tokens' in request.headers:  
            token = request.headers['x-access-tokens']  
        if not token:  
            return jsonify({'message': 'a valid token is missing'})  
  
        data = jwt.decode(token, config.SECRET_KEY, algorithms=["HS256"])  
        print("data: {}".format(data))  
        current_user = Users.query.filter_by(public_id=data['public_id']).first()  
        print("current user: {}".format(current_user.name))  
        return f(current_user, *args, **kwargs)  
    return decorator
```

```
@app.route('/users', methods=['GET'])  
@token_required  
def get_all_users(current_user):  
    users = Users.query.all()
```

Harness analytical strengths: SAS, R, Python Web Services

Securing Endpoints – Implementing HTTPS

- Generate Certificate and key by following instructions for Method 2 from - <https://kracekumar.com/post/54437887454/ssl-for-flask-local-development/>
 - Generate a private key `openssl genrsa -des3 -out server.key 1024`
 - Generate a CSR `openssl req -new -key server.key -out server.csr`
 - Remove Passphrase from key `cp server.key server.key.org openssl rsa -in server.key.org -out server.key`
 - Generate self signed certificate `openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt`



server.crt
server.csr
server.key
server.key.org

```
server = WSGIServer(bind_addr=("0.0.0.0", int(5000)), wsgi_app=app, numthreads=100)
server.ssl_adapter = pyOpenSSLAdapter("./certs/server.crt", "./certs/server.key", None)
```

Harness analytical strengths: SAS, R, Python Web Services

Next Steps

- Webservice vs Microservice

- a web service is (as the name announces) something that you would expect to "deal" with HTTP - in other words it does something in the context of the world wide web.
- Whereas a microservice is not subject to the WWW context. At its core a microservice is supposed to provide one specific service - but there is no restriction to the http protocol for example.

Reference: <https://stackoverflow.com/questions/45453061/what-is-the-difference-between-microservices-and-webservices/45453646>

- VM vs Docker Image
- Docker Compose vs Kubernetes
- Onsite Hosting vs Cloud Hosting

Questions

Contact: Sandeep.Juneja@sas.com



sas.com

