Eventgesteuerte Architektur mit SAS® 9.4

Senad Jukic

Caputure the Events..

We just want to..

- Run some SAS code
- Create a report
- Merge files

Caputure the Events..

We just want to..

- Run some SAS code
- Create a report
- Merge files

When..

- A state changes (new file, new database entry, ..)
- Every hour
- Certain conditions are met
- Git merge is approved

Go Serverless!

Events: created when database, file system, or object storage changes, or API gateways were triggered (e.g. HTTP call)

Triggers: can be set to listen to events and be attached to functions

Functions: piece of code that runs in containers that are being invoked by triggers. Container runtime is handled by public cloud provider and billed upon invocation.

Go Serverless!

Where?

- Azure: Azure Functions
- AWS: AWS Lamda
- GCP: Cloud Functions
- Private Datacenter: faasd

How?

- Portal + ZIP Upload
- Provider CLI's
- Serverless Framework
- Shell scripts accessing REST API

Go Serverless!

Where?

- Azure: Azure Functions
- AWS: AWS Lamda
- GCP: Cloud Functions
- Private Datacenter: faasd

How?

- Portal + ZIP Upload
- Provider CLI's
- Serverless Framework
- Shell scripts accessing REST API

Go Serverless!

SAS9 Interfaces?

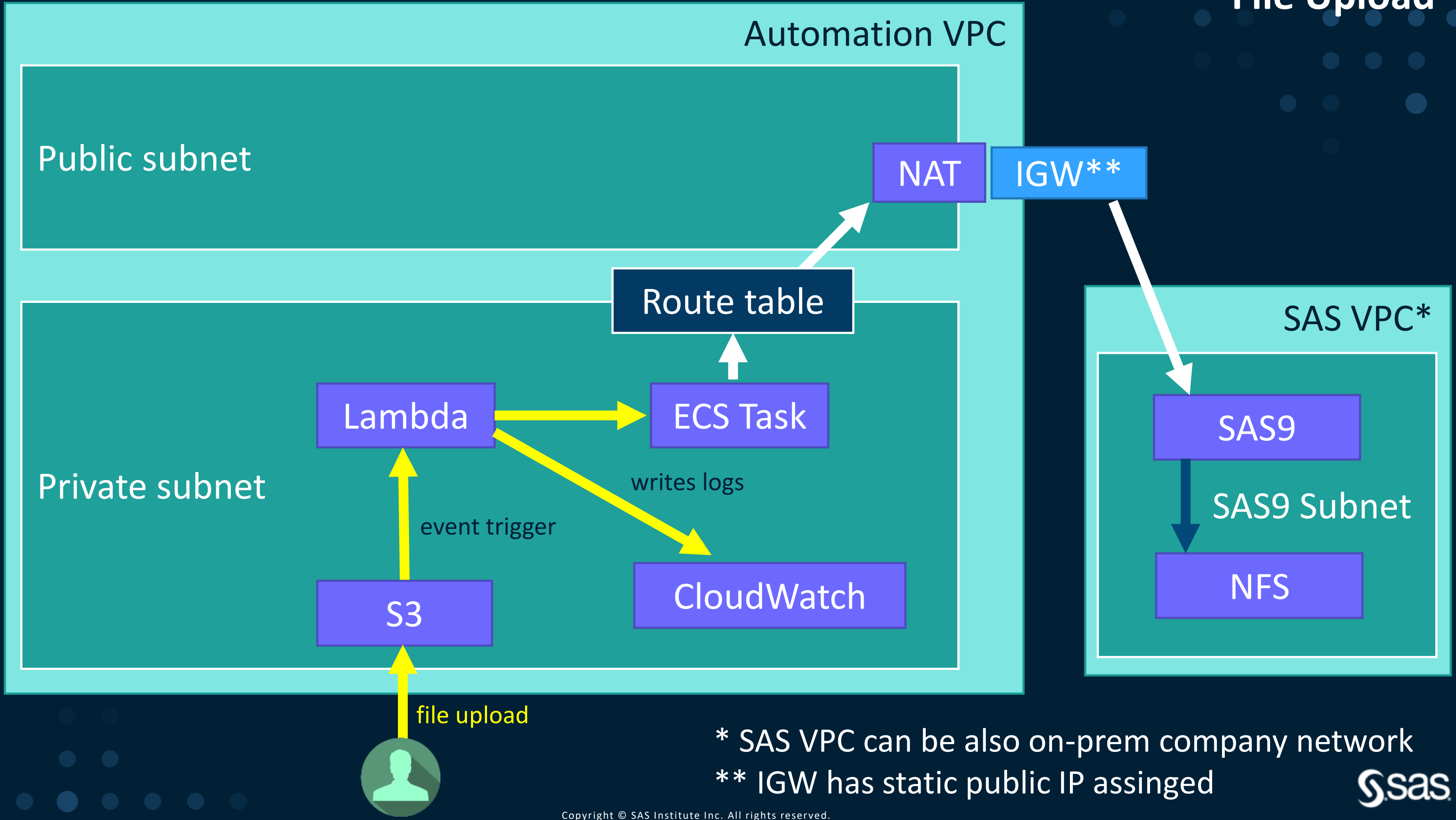
- SASpy (Python Client Library)
- SSH Remote
  - Uses Pseudo TTY, which is not allowed by for example AWS Lambda

https://sassoftware.github.io/saspy/configuration.html#remote

Therefore, use Serverless Compute Engine (ECS) instead Lambda in combination with SAS9.

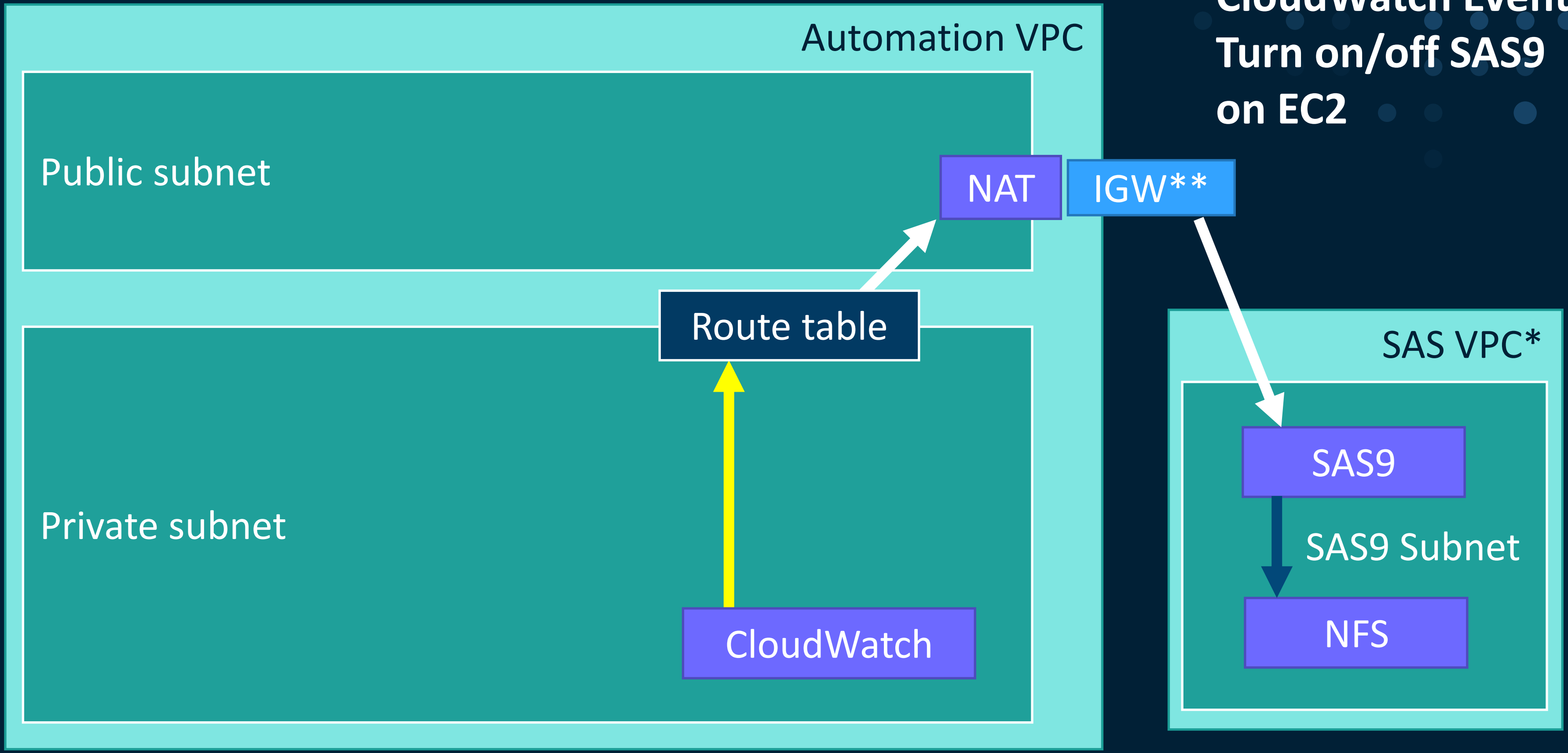
ECS can be based on Fargate or EC2.

File Upload



* SAS VPC can be also on-prem company network
** IGW has static public IP assinged

CloudWatch Event Turn on/off SAS9 on EC2



* SAS VPC can be also on-prem company network

** IGW has static public IP assinged

Triggers (AWS Example)

- Elastic Load Balancing (Application Load Balancer)
- Amazon Cognito
- Amazon connect
- Amazon Lex
- Amazon Alexa
- Amazon API Gateway
- Amazon CloudFront (Lambda@Edge)
- Amazon Kinesis Data Firehose
- Amazon Simple Storage Service (S3)
- Amazon Simple Notification Service
- Amazon Simple Email Service
- AWS CloudFormation
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- AWS CodeCommit
- AWS Config
- AWS IoT
- AWS IoT Events
- AWS CodePipeline

Advanced Usecase

- Store SAS programs + data together on S3 buckets
- If there is a change to code or data, react
- Make it simple for users to publish on S3
- Do not run all the SAS code if not needed

Dependency Checker via Data Lineage

Collect SAS programs Input(s) / Output(s)

_run_batch.sas × master_io × +

master_io

Table Rows: 106 | Columns: 9 of 9 | Rows 1 to 12 (filtered)

sas_task="/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas"

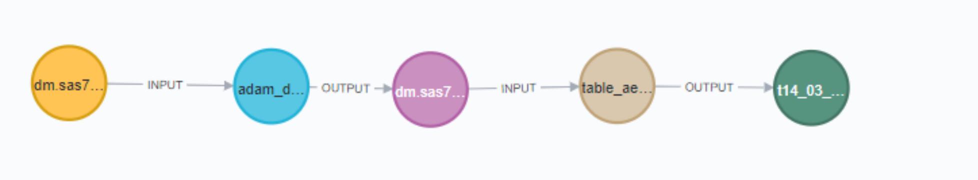
	sas_task	order...	type	obj_type	file_path	timeinfo
1	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	1	JOBSTART...	JOBSTARTTIME	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	21JUN2021:17:46:58.16
2	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	2	JOBENDTIME	JOBENDTIME	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	21JUN2021:17:46:58.48
3	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/rawdata/dm.sas7bdat	
4	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/rawdata/ds.sas7bdat	
5	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/rawdata/oe.sas7bdat	
6	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/rawdata/sda.sas7bdat	
7	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/rawdata/spc.sas7bdat	
8	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/rawdata/vs.sas7bdat	
9	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_004/macros/autoexec_viya.sas	
10	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	3	INPUT	FILE	/compute-landingzone/Projects/hlsdata/study_004/macros/tplate.sas7bitm	
11	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	4	OUTPUT	FILE	/compute-landingzone/Projects/hlsdata/study_003/data/adam/dm.sas7bdat	
12	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/adam_d0_dm.sas	4	OUTPUT	FILE	/compute-landingzone/Projects/hlsdata/study_004/programs/dev/logs/adam_d	

Last_Modified	File_Size__bytes
21Jun2021:13:43:52	5717
21Jun2021:13:43:52	5717
02Jun2021:18:06:28	17408
02Jun2021:18:06:28	50176
02Jun2021:18:06:28	13312
02Jun2021:18:06:28	66560
02Jun2021:18:06:28	33792
02Jun2021:18:06:28	25600
21Jun2021:13:43:51	2674
21Jun2021:13:43:51	40960
21Jun2021:13:46:58	163840
21Jun2021:13:46:58	19555

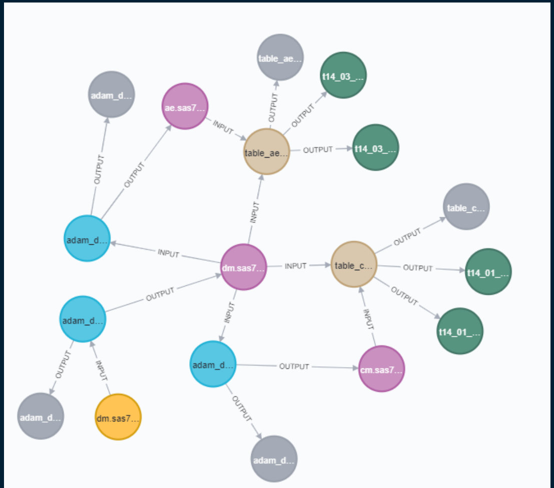
- sas_task – Executing SAS program
- Type – JobStartTime/JobEndTime/Input/Output
- file_path – referenced file by executing sas program
- last_modified - last modified datetime
- File_size_bytes – file size in bytes

Data Lineage

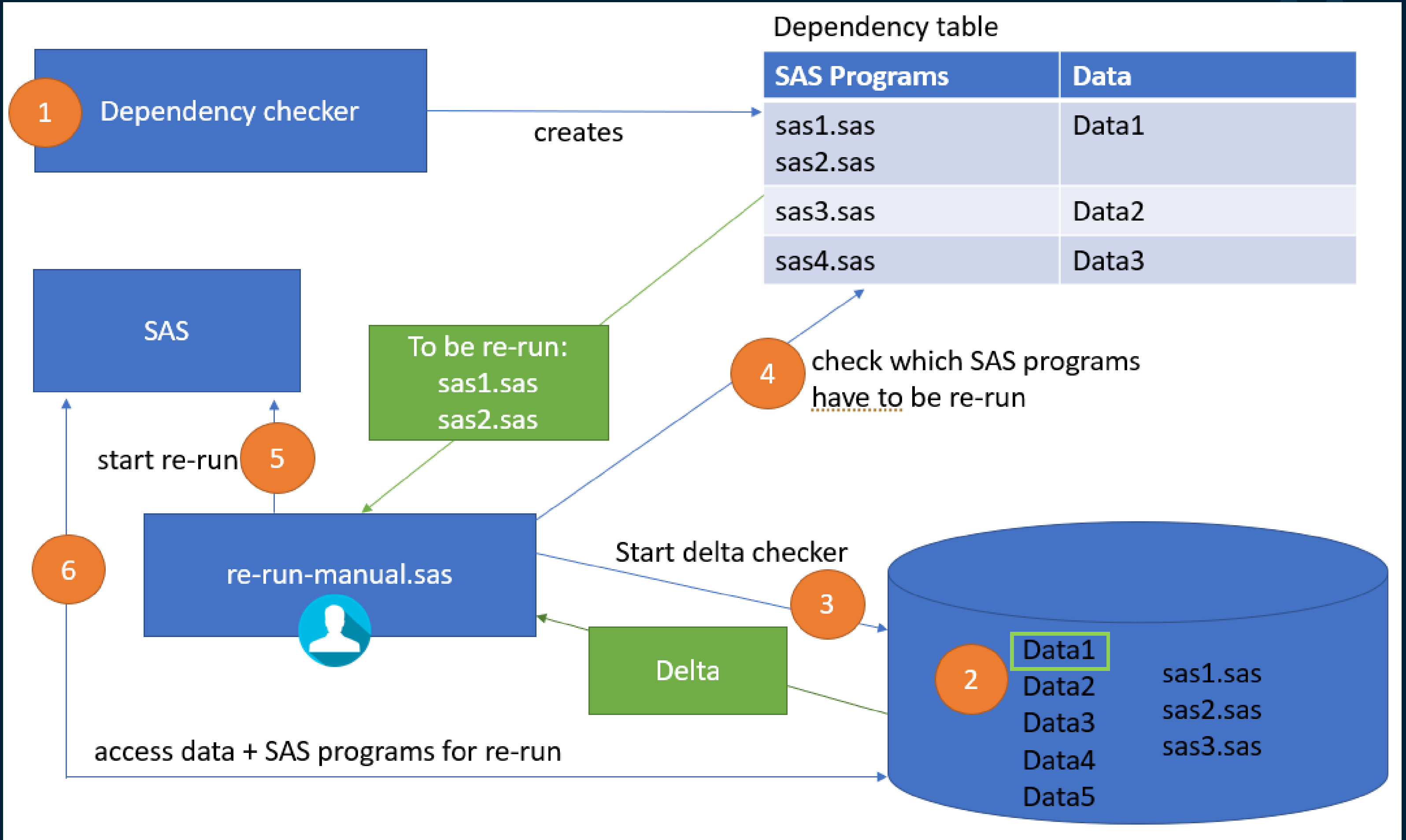
Simplified



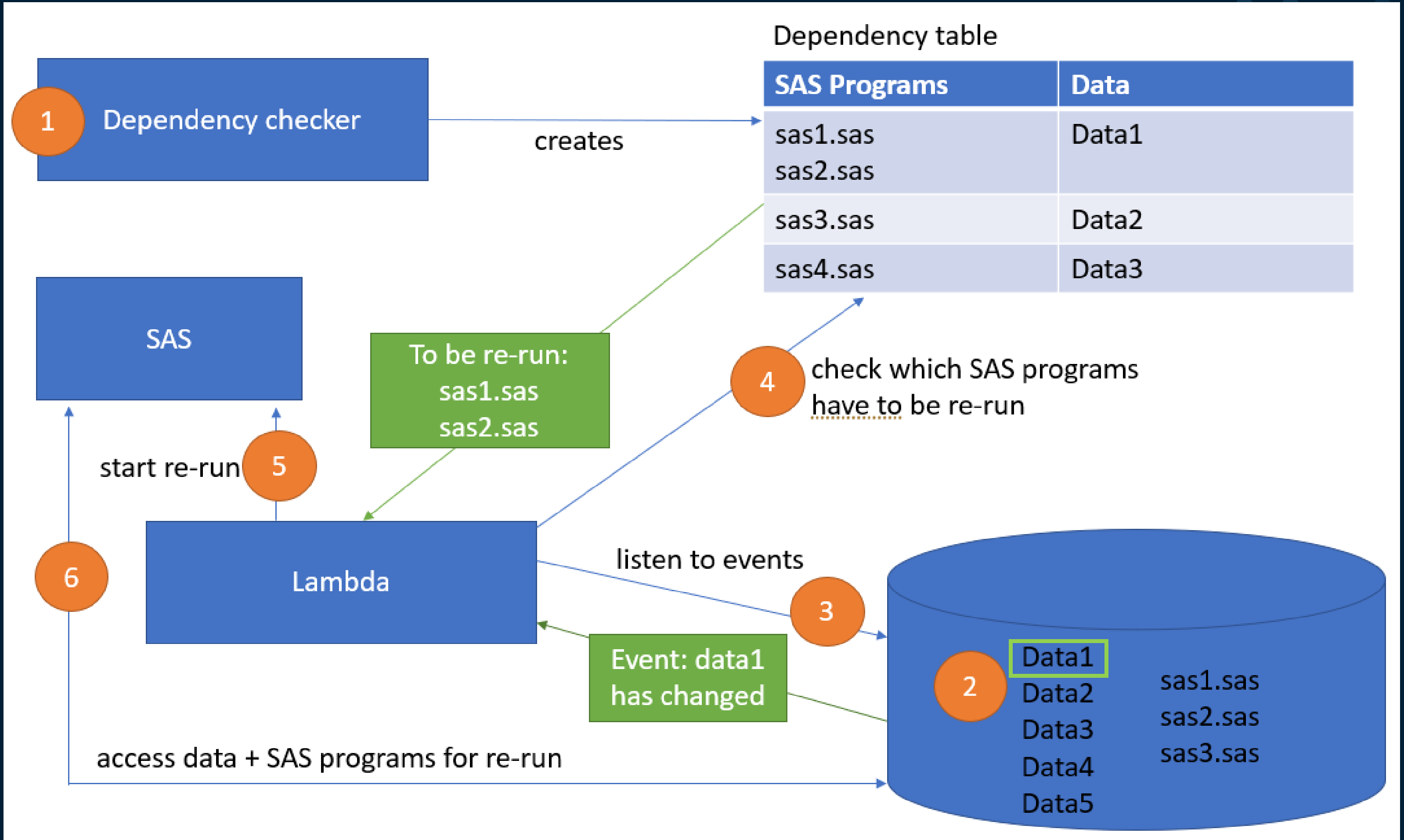
Direct / Indirect dependency



Manual Partial Execution on Data/Program Change

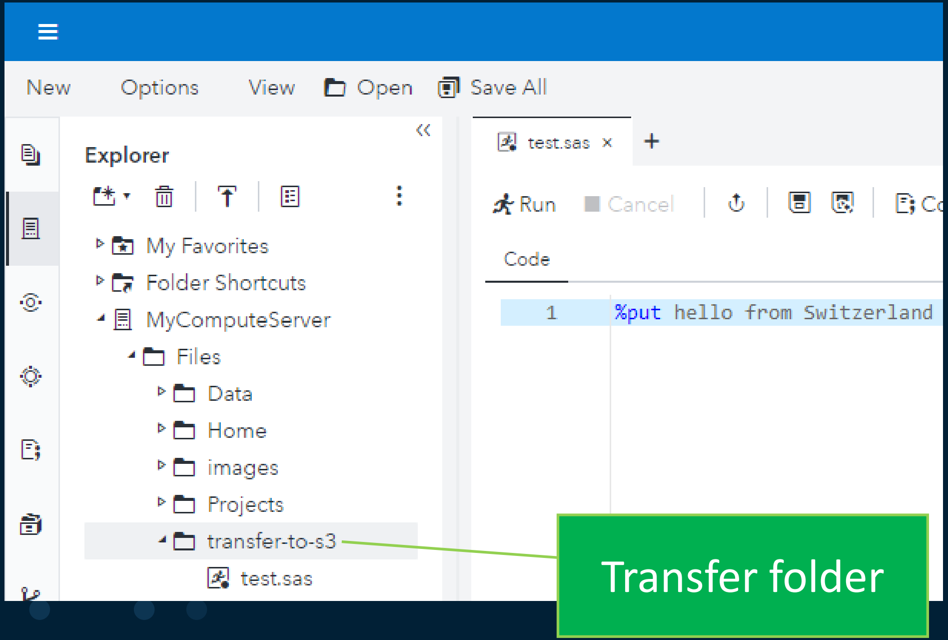


Automated Partial Execution on Data/Program Change



Auto Transfer from EFS to AWS S3

Scenario: use automated transfer folders to transfer data&scripts from EFS to S3, e.g. for archiving final run results



Each file or folder put into that transfer folder will be archived on S3.

Each 30 seconds, the sync checker is executed.

Flow:

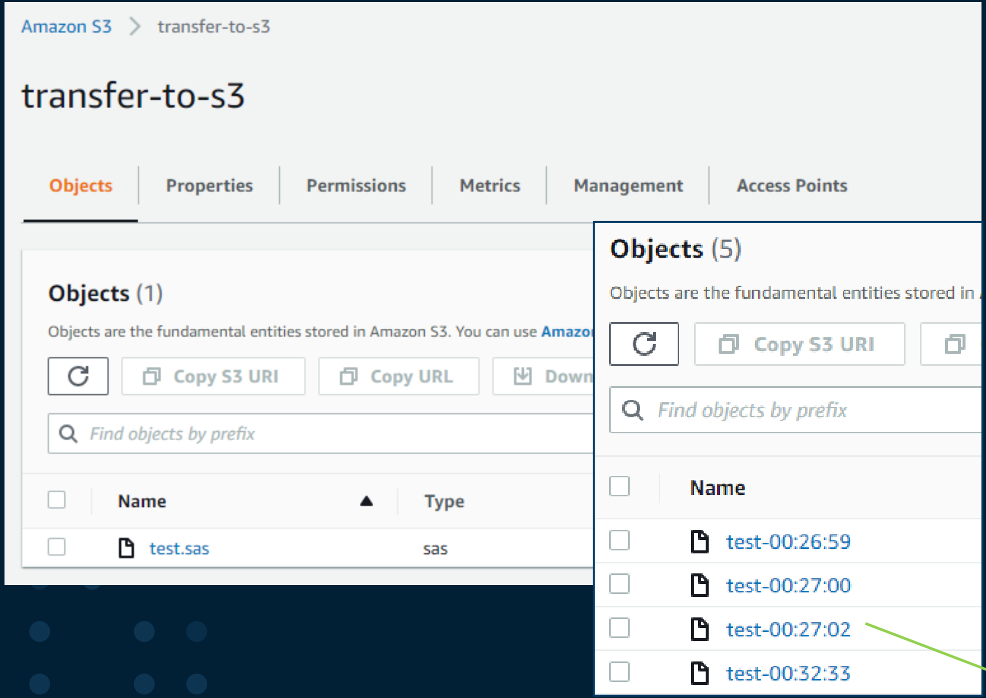
1) SAS User puts file/folder to transfer folder
2) Sync is being executed all 30 seconds
3) Files in subfolder will be deleted

Naming logic can be adjusted, e.g. add timestamp to folder names to avoid overwriting.

This is a one-way sync. Deletion on EFS is not affecting the S3!

Auto Transfer from EFS to AWS S3

Scenario: use automated transfer folders to transfer data&scripts from EFS to S3, e.g. for archiving final run results



Each file or folder put into that transfer folder will be archived on S3.

Each 30 seconds, the sync checker is executed.

Flow:

1) SAS User puts file/folder to transfer folder
2) Sync is being executed all 30 seconds
3) Files in subfolder will be deleted

Naming logic can be adjusted, e.g. add timestamp to folder names to avoid overwriting.

Each file/folder can contain timestamp

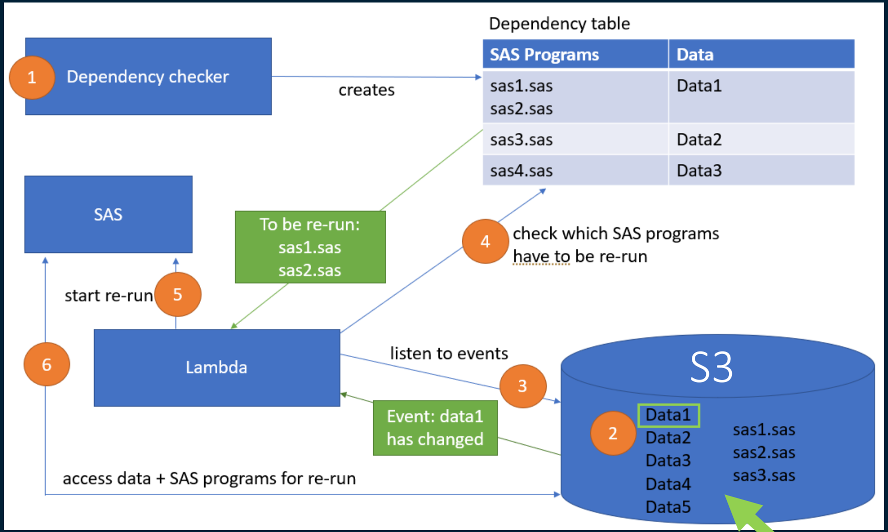
Auto Transfer from EFS to AWS S3

transfer-loop.sh

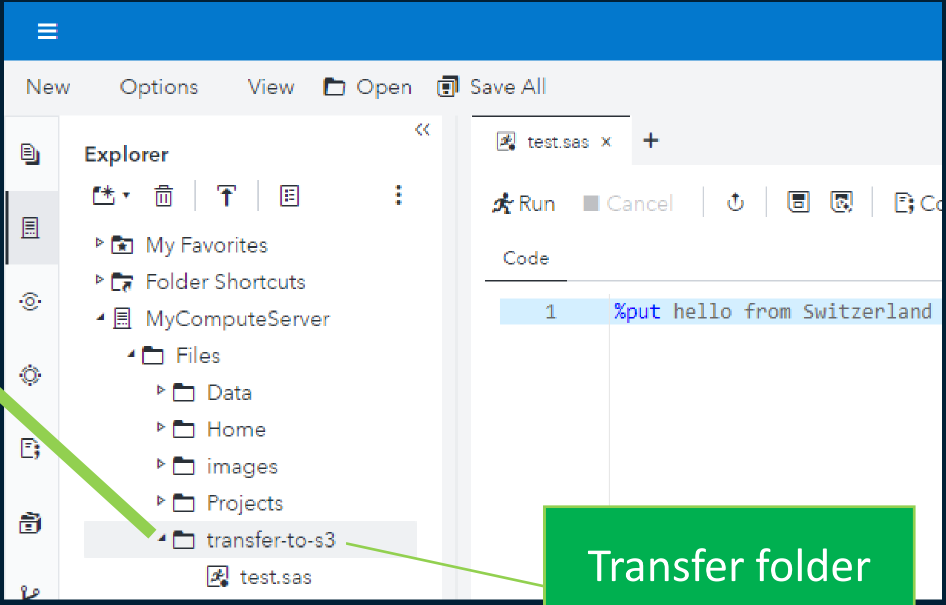
```
#!/bin/bash
while true
do
  if [ $(ls -A "/transfer-to-s3" | wc -l) -ne 0 ]; then
    aws s3 sync /transfer-to-s3 s3://transfer-to-s3 > /dev/null
    rm -rf /transfer-to-s3/*
  fi
  sleep 30
done
```

```
# prepare and setup background process
chmod +x transfer-loop.sh
nohup ./transfer-loop.sh &
```

Auto Transfer from EFS to AWS S3



..with automated partial execution on data/program change



Advanced Usecase

- ✓ Store SAS programs + data together on S3 buckets
- ✓ If there is a change to code or data, react
- ✓ Make it simple for users to publish on S3
- ✓ Do not run all the SAS code if not needed

Thank you

Senad Jukic
Cloud Architect
senad.jukic@sas.com