# AWS Lambda, Cognito, S3

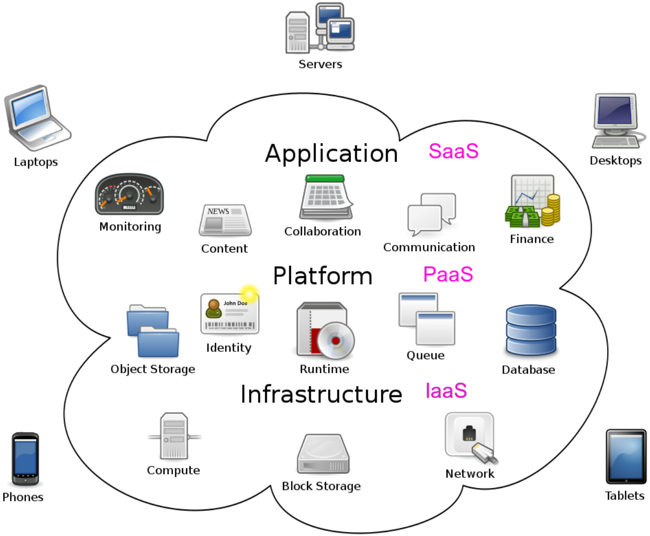
## In 3 days from idea to a working solution with AWS Lambda, Cognito, S3

Vladimir Dobriakov

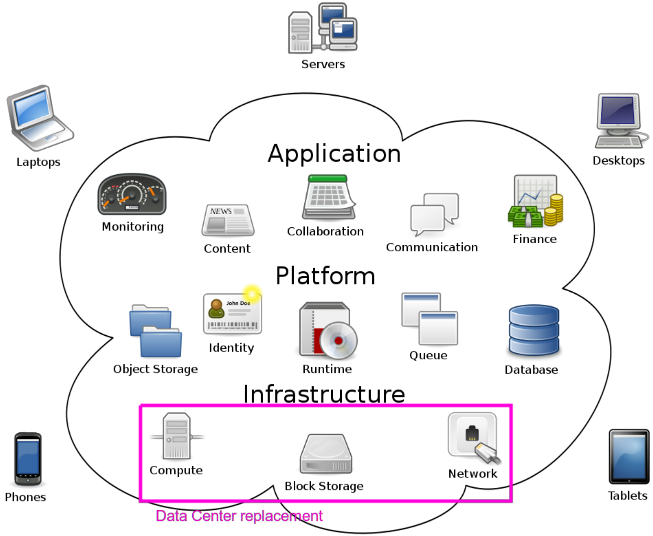
http://infrastructure-as-code.de

AWS UG Cologne 2023-11-30

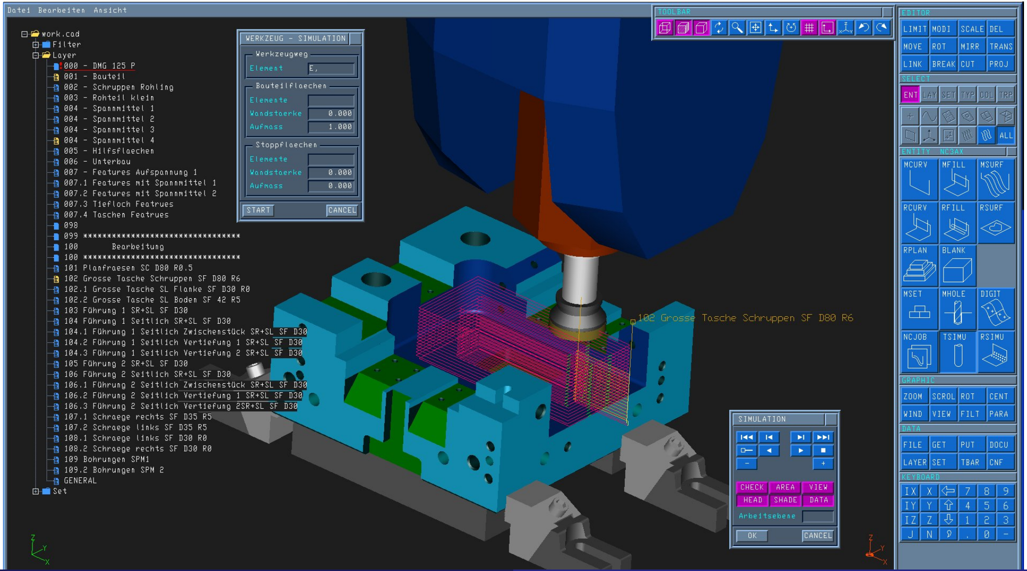
# Cloud Computing



# Glorified Data Center



# Our Use Case - deliver CAD / CAM / MES software



```
Datei  Bearbeiten  Ansicht

work.cad
  Filter
  Layer
    000 - DMG 125 P
    001 - Bauteil
    002 - Schruppen Rohling
    003 - Rohteil klein
    004 - Spannmittel 1
    004 - Spannmittel 2
    004 - Spannmittel 3
    004 - Spannmittel 4
    005 - Hilfsflaechen
    006 - Unterbau
    007 - Features Aufspannung 1
    007.1 Features mit Spannmittel 1
    007.2 Features mit Spannmittel 2
    007.3 Tiefloch Featrues
    007.4 Taschen Featrues
    098
    099 ***********************************
    100       Bearbeitung
    100 ***********************************
    101 Planfraesen SC D80 R0.5
    102 Grosse Tasche Schruppen SF D80 R6
    102.1 Grosse Tasche SL Flanke SF D30 R0
    102.2 Grosse Tasche SL Boden SF 42 R5
    103 Führung 1 SR+SL SF D30
    104 Führung 1 Seitlich SR+SL SF D30
    104.1 Führung 1 Seitlich Zwischenstück SR+SL SF D30
    104.2 Führung 1 Seitlich Vertiefung 1 SR+SL SF D30
    104.3 Führung 1 Seitlich Vertiefung 2 SR+SL SF D30
    105 Führung 2 SR+SL SF D30
    106 Führung 2 Seitlich SR+SL SF D30
    106.1 Führung 2 Seitlich Zwischenstück SR+SL SF D30
    106.2 Führung 2 Seitlich Vertiefung 1 SR+SL SF D30
    106.3 Führung 2 Seitlich Vertiefung 2SR+SL SF D30
    107.1 Schraege rechts SF D35 R5
    107.2 Schraege links SF D35 R5
    108.1 Schraege links SF D30 R0
    108.2 Schraege rechts SF D30 R0
    109 Bohrungen SPM1
    109.2 Bohrungen SPM 2
    GENERAL
  Set
```

WERKZEUG - SIMULATION

Werkzeugweg
Element: E,

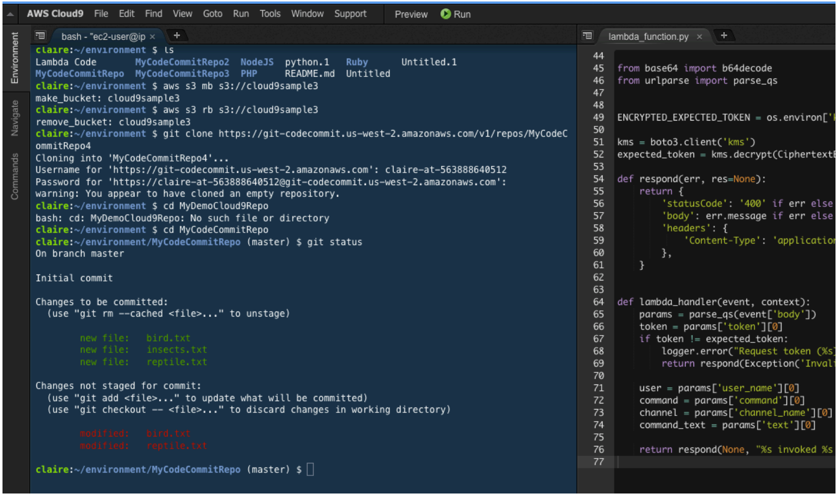
Bauteilflaechen
Elemente:
Wandstaerke: 0.000
Aufmass: 1.000

Stoppflaechen
Elemente:
Wandstaerke: 0.000
Aufmass: 0.000

102 Grosse Tasche Schruppen SF D80 R6

SIMULATION
Arbeitsebene:

# No admin rights? Try Cloud9

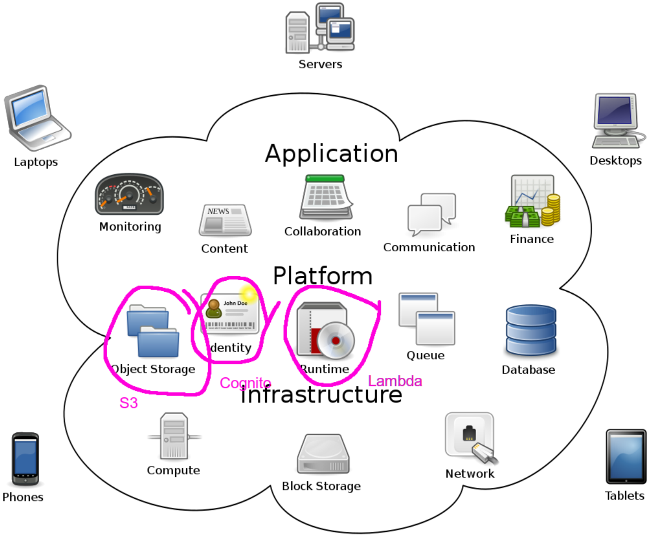


```
claire:~/environment $ ls
Lambda Code        MyCodeCommitRepo2  NodeJS  python.1   Ruby      Untitled.1
MyCodeCommitRepo   MyCodeCommitRepo3  PHP     README.md  Untitled
claire:~/environment $ aws s3 mb s3://cloud9sample3
make_bucket: cloud9sample3
claire:~/environment $ aws s3 rb s3://cloud9sample3
remove_bucket: cloud9sample3
claire:~/environment $ git clone https://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyCodeC
ommitRepo4
Cloning into 'MyCodeCommitRepo4'...
Username for 'https://git-codecommit.us-west-2.amazonaws.com': claire-at-563888640512
Password for 'https://claire-at-563888640512@git-codecommit.us-west-2.amazonaws.com':
warning: You appear to have cloned an empty repository.
claire:~/environment $ cd MyDemoCloud9Repo
bash: cd: MyDemoCloud9Repo: No such file or directory
claire:~/environment $ cd MyCodeCommitRepo
claire:~/environment/MyCodeCommitRepo (master) $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   bird.txt
        new file:   insects.txt
        new file:   reptile.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   bird.txt
        modified:   reptile.txt

claire:~/environment/MyCodeCommitRepo (master) $ 
```

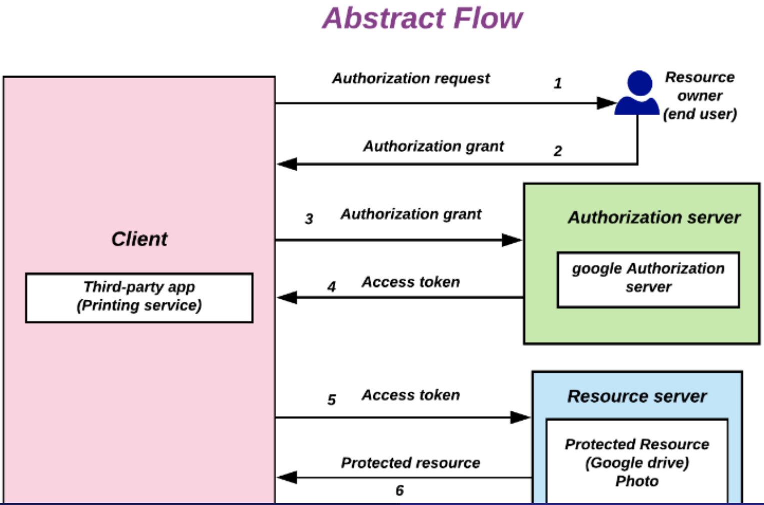
```python

from base64 import b64decode
from urlparse import parse_qs


ENCRYPTED_EXPECTED_TOKEN = os.environ['

kms = boto3.client('kms')
expected_token = kms.decrypt(Ciphertext

def respond(err, res=None):
    return {
        'statusCode': '400' if err else
        'body': err.message if err else
        'headers': {
            'Content-Type': 'applicatio
        },
    }


def lambda_handler(event, context):
    params = parse_qs(event['body'])
    token = params['token'][0]
    if token != expected_token:
        logger.error("Request token (%s
        return respond(Exception('Inval

    user = params['user_name'][0]
    command = params['command'][0]
    channel = params['channel_name'][0]
    command_text = params['text'][0]

    return respond(None, "%s invoked %s

```

Figure 4: Cloud 9

# Technologies for Our Use Case



# OAuth2


Abstract Flow
Authorization request
1
Resource owner (end user)
Authorization grant
2
Client
Third-party app (Printing service)
3
Authorization grant
Authorization server
google Authorization server
4
Access token
5
Access token
Resource server
Protected Resource (Google drive) Photo
Protected resource
6

# Source Code

**Dive into Source Code**

# SAM Template -> CloudFormation Template

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  DownloadPortalFunction:
    Type: AWS::Serverless::Function
    DependsOn:
      - UserPool
      - UserPoolClient
    Properties:
      PackageType: Image
      Environment:
        Variables:
          USERPOOL_ID: !Ref UserPool
          CLIENT_ID: !Ref UserPoolClient
          CLIENT_SECRET: !Ref AppClientSecret
          LOGIN_DOMAIN: !Ref UserPoolDomain
          S3_BUCKET: !Ref S3DownloadBucket
```

# SAM Template - Lambda

```
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /download
            Method: get
      Policies:
        - S3ReadPolicy:
            BucketName:
              !Ref S3DownloadBucket
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./lambda_auth
      DockerTag: python3.9-v1
```

# Add your own resources, specific for your solution

```
UserPool:
  Type: AWS::Cognito::UserPool
  Properties:
    #UserPoolName: MyUserPool
    UsernameAttributes:
      - email
    Policies:
      PasswordPolicy:
        MinimumLength: 8
    Schema:
      - AttributeDataType: String
        Name: email
      - AttributeDataType: String
        Name: ”Order_Number”
```

# Configure UserPool with OAuth

```
  UserPoolClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      CallbackURLs:
        - !Sub "https://${PortalURLPrefix}.execute-api.${AWS::Region}.amazo
      # - !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazo
      AllowedOAuthFlowsUserPoolClient: True
      GenerateSecret: True
      AllowedOAuthScopes:
        - email
        - openid
        - profile
      SupportedIdentityProviders:
        - COGNITO
      AllowedOAuthFlows:
```

# Add Users

Add known users including additional attributes like order number

```
Outputs:
  AddCognitoUser:
    Description: ”Call to create cognito default user”
    Value: !Sub ”aws cognito-idp admin-create-user --user-pool-id
        ${UserPool}
        --username max.mustermann@infrastructure-as-code.de
        --user-attributes
            Name=email,Value=max.mustermann@infrastructure-as-code.de
      Name=custom:CO_Number,Value=CO-0099999”
```

# S3 Bucket

```
  S3DownloadBucket:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: Private

Outputs:
  AddS3BucketContent:
    Value: !Sub ”aws s3 cp licence.key
        s3://${S3DownloadBucket}/CO−005176/”
```

# Questions

**Questions ???**

# Bildnachweis

- Cloud_computing - Sam Johnston, CC BY-SA 3.0, via Wikimedia Commons
- CAD/CAM - Tebis Technische Informationssysteme AG, CC BY-SA 3.0, via Wikimedia Commons
- AWS cloud9 - screenshot AWS product description page
- OAuth 2.0 flow - Devansvd, CC-BY-SA-4.0, via Wikimedia Commons