

Welcome
**Platform &
Pizzas**

Personio



Pierre Piriou
Engineering
Manager



Alexey Miroshnikov
Lead Engineer



Meet our team tonight

Philipp Garbe
Lead Engineer



Andreas Sieferlinger
Senior Site Reliability Engineer





Join our Tech Community Newsletter

Join our Tech-Community!

- Get pictures & videos from this event.
- Get invites to more networking events and meetups at Personio offices.
- Industry insights and learnings from our hottest tech blogs.
- Updates on our newest Product & Technology open roles.



aws

Cloud
Development
Kit

Infrastructure as Code for a fast growing Engineering Organization

Personio

Andreas Sieferlinger
Senior Site Reliability Engineer

Agenda

01 Intro

02 A brief history of Infra as Code

03 Extending CDK

04 Dealing with compliance

05 Q&A



Infrastructure as Code

How the setup looked like initially

How it started

- Few centrally managed terraform repos
- Split out from initial huge monolithic terraform repo
- Contribution via Merge Request
- Low number of teams have own terraform repos

Issues we faced

- Review and deployment becomes bottleneck over time
- Terraform / HCL unknown to people outside of infrastructure
- Many very similar modules for similar tasks - confusion

Engineering at Personio

~45 Teams in PTECH (Engineering Organization), ~6 Teams in TPF (Tech Platform)

Brief IaC History Walkthrough

02

Personio



Manual / UI driven

- 👍 Easy to get started
- 😞 Not reproducible
- 😞 Error prone
- 😞 Time consuming

High
level



Low
level

Manual

Scripted

- 👍 Fast
- 😞 Handle failed API calls
- 😞 How to update resources?
- 😞 When is a resource ready?
- 😞 How to do a rollback?

High
level



Low
level

Scripted

Manual

```
55
56     if not db_type:
57         raise NoDatabaseFoundError(
58             f'could neither find a cluster nor an instance for {db_id}')
59
60     def get_latest_snapshot(self) -> str:
61         if self.snapshot_id:
62             return self.snapshot_id
63
64         latest_snapshot = {'SnapshotCreateTime': datetime.datetime(
65             1970, 1, 1, tzinfo=datetime.timezone.utc)}
66         for page in self.rds.get_paginator('describe_db_snapshots').paginate(DBInstanceIdentifier=self.source_db_id):
67             for snapshot in page['DBSnapshots']:
68                 if snapshot['SnapshotCreateTime'] > latest_snapshot['SnapshotCreateTime']:
69                     latest_snapshot = snapshot
70
71         try:
72             return latest_snapshot['DBSnapshotIdentifier']
73         except KeyError:
74             raise NoSnapshotFoundError(
75                 f'could not find latest snapshot. only works for instances')
76
77     def get_cluster_configuration(self):
78         clusters = self.rds.describe_db_clusters(
79             DBClusterIdentifier=self.source_db_id)['DBClusters']
80         if len(clusters) > 1:
81             raise TooManyClustersError(
82                 f'more than one db cluster found for {self.source_db_id}')
83
84         config = {
85             'DBClusterIdentifier': self.target_db_id,
86             'SnapshotIdentifier': self.get_latest_snapshot(),
87             'Engine': clusters[0]['Engine'], # 'aurora-postgresql',
88             'EngineVersion': clusters[0]['EngineVersion'],
89             'EngineMode': clusters[0]['EngineMode'],
90             'DBSubnetGroupName': clusters[0]['DBSubnetGroup'],
91             'DBClusterParameterGroupName': clusters[0]['DBClusterParameterGroup'],
92             'VpcSecurityGroupIds': [sg['VpcSecurityGroupId'] for sg in clusters[0]['VpcSecurityGroups']]
93         }
94         return config
95
96     def get_instance_configuration(self):
97         instances = self.rds.describe_db_instances(
98             DBInstanceIdentifier=self.source_db_id)['DBInstances']
99         if len(instances) > 1:
100             raise TooManyInstancesError(
101                 f'more than one db instance found for {self.source_db_id}')
102
103         config = {
104             'DBInstanceIdentifier': self.target_db_id,
105             'DBSnapshotIdentifier': self.get_latest_snapshot(),
106             'DBSubnetGroupName': instances[0]['DBSubnetGroup']['DBSubnetGroupName'],
107             'DBParameterGroupName': instances[0]['DBParameterGroups'][0]['DBParameterGroupName'],
108             'VpcSecurityGroupIds': [sg['VpcSecurityGroupId'] for sg in instances[0]['VpcSecurityGroups']]
109         }
110         return config
111
112     def get_config_from_file(self, filepath):
113         try:
114             with open(filepath) as f:
```

Resource Provisioning Engines

- 👍 Easy to automate
- 👍 Reproducible
- 😞 Error prone
- 😞 Time consuming
- 😞 Hard to test

High
level

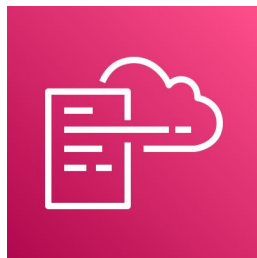


Low
level

Declarative

Scripted

Manual



Cloudformation
JSON / YAML

HCL



Terraform

Generators

Object Document Models (DOMs)

- 👍 Real code
- 👍 Desired state
- 😞 No abstractions
- 😞 Time consuming

High level



Low level

Generators

Declarative

Scripted

Manual

Troposphere (python)
SparkleFormation (ruby)
GoFormation (go)

```
1 from troposphere import Parameter, Ref, Template
2 from troposphere.ecs import (
3     AwsVpcConfiguration,
4     Cluster,
5     ContainerDefinition,
6     NetworkConfiguration,
7     PortMapping,
8     Service,
9     TaskDefinition,
10 )
11
12 t = Template()
13 t.set_version("2018-09-09")
14 t.add_parameter(
15     Parameter(
16         "Subnet",
17         Type="AWS::EC2::Subnet::Id",
18         Description="A VPC subnet ID for the container.",
19     )
20 )
21
22 cluster = t.add_resource(Cluster("cluster"))
23
24 task_definition = t.add_resource(
25     TaskDefinition(
26         "TaskDefinition",
27         RequiresCompatibilities=["FARGATE"],
28         Cpu="256",
29         Memory="512",
30         NetworkMode="awsvpc",
31         ContainerDefinitions=[
32             ContainerDefinition(
33                 Name="nginx",
34                 Image="nginx",
35                 Essential=True,
36                 PortMappings=[PortMapping(ContainerPort=80)],
37             )
38         ],
39     )
40 )
41
42 service = t.add_resource(
43     Service(
44         "NginxService",
45         Cluster=Ref(cluster),
46         DesiredCount=1,
47         TaskDefinition=Ref(task_definition),
48         LaunchType="FARGATE",
49         NetworkConfiguration=NetworkConfiguration(
50             AwsVpcConfiguration=AwsVpcConfiguration(Subnets=[Ref("Subnet")])
51         ),
52     )
53 )
54
55 print(t.to_json())
56
```

Abstractions

- 👍 Composable constructs
- 👍 Less code to write
- 👍 Good to test
- 🤔 More complexity

High
level



Low
level

Abstractions

Generators

Declarative

Scripted

Manual



aws
CDK

AWS CDK - *typescript*
pulumi

Why CDK?

- Well integrated with AWS
- Active open source project
- No additional commercial interest
- Proven experience with rollout to bigger engineering organizations
- Stable underlying interface (CloudFormation)

Goals of CDK usage

- Empower development teams to manage their own infrastructure
 - No more waiting for infra MR approvals
 - Infrastructure in same repo as application code
 - Faster feedback cycles while developing
 - Better quality of infrastructure through tests and reusable constructs
- Easier to get started: Standard Programming language instead of HCL / CloudFormation

Extending CDK

03

Personio



Bootstrapping CDK Infra

Out of the Box

- cdk bootstrap command needed
- cdk role permissions need to be set
- Custom resources for additional services need to be deployed and configured

Personio flavoured CDK setup

- Bootstrap Stack preconfigured in every AWS Account
- Datadog custom resources pre-deployed and configured

Project setup

Out of the Box

- Blank or projen

Personio flavoured CDK

- Cookiecutter for full project
- Projen for CDK setup
 - Updates via library updates
 - Files generated from Typescript
- CI Pipeline preconfigured

Creating project from template



```
1 cookiecutter https://gitlab.internal/templates/microservice-template-kotlin
```



```
1 cd personio-cdk-kotlin-example  
2 cd ops/cdk  
3 yarn install  
4 yarn cdk --profile dev deploy --all
```

Creating project from template

```

  ops
  cdk
    .projen
    cdk.out
    node_modules
  src
    TS main.ts
    TS service-stack.ts
  test
    TS service-stack.test.ts
.eslintrc.json
.gitattributes
.gitignore
.npmignore
.npmrc
.projenrc.ts
cdk-outputs.json
cdk.context.json
cdk.json
package.json
README.md
tsconfig.dev.json
tsconfig.json
yarn.lock
```

Integration with Infrastructure

Out of the Box

- Use fromLookup methods
 - But what value to put?

Personio flavoured CDK

- Write SSM Parameter
- Use fromAttributes / fromLookup - predefined
- VPC
- DNS
- KMS
- Opsgenie Alerting
- Tagging
- Metadata added

Example of base Infra usage



```
1 const kms = BaseKMS.getDataKeybyAlias(this);  
2  
3 const zone = BaseDNS.getPersonioInternalHostedZone(this);
```

Construct ... what?

A building Block within CDK

- Encapsulates a “cloud component”
- Great to provide opinionated defaults
- Can consist of a single or many resources
- Can easily interact with other constructs

Opinionated custom constructs

Tenets for custom constructs

- Compliant
- Secure
- Flexible
- Operations built in
 - Alarms
 - Helpers
 - Dashboards / Widgets
- Well tested
- Documentation with examples

Subset of provided constructs

- Aurora postgres
- ElastiCache Redis
- ECS Fargate
- DynamoDB
- SNS
- SQS
- ...


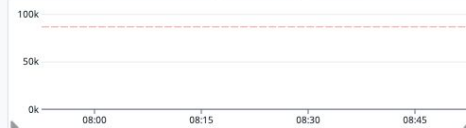
Opinionated custom constructs



```
1 const queue = new sqs.SQSQueue(this, 'serviceQueue', {  
2     opsgenieTeamName: 'TPF_IE_CI',  
3     enableDeadLetterQueue: true  
4 });
```

Add  Widgets or  Powerpacks

AWS Cost Explorer

service/serviceQueue/AlarmApproximateAgeOfOlde...    service/serviceQueue/ApproximateNumberOfMessagesVisi...    service/serviceQueue/ApproximateNumberOfMessagesVisi...    

Opinionated custom constructs



```
1 const queue = new sqs.SQSQueue(this, 'serviceQueue', {
2     opsgenieTeamName: props.opsgenieTeamName,
3     enableDeadLetterQueue: true,
4     alarmOverrides: {
5         approximateAgeOfOldestMessage: {
6             alarm: {
7                 threshold: cdk.Duration.hours(6).toSeconds(),
8             },
9         },
10     },
11 });
```

**Staying compliant - at
full speed**

04

Personio



Compliance challenges

- We deal with PII data almost all the time
- GDPR / BDSG applies
- Requirements from Certifications
- Multi tenant setup
- Engineers need access to some data for debugging

Compliance challenges

- We deal with PII data almost all the time
- GDPR / BDSG applies
- Requirements from Certifications
- Multi tenant setup
- Engineers need access to some data for debugging




Detecting risky configuration

Out of the Box

- `cdk diff`

Personio flavoured CDK

- `cdk-nag` 
- Using set of default rules
- Extending by own rules from SEC team
- Additional: AWS Config for deployed resources

Autonomy & self service

Removing Processes and Empowering Employees: Personio Empower

**Could you explain in the company wide
All hands why this is useful for Personio?**

**Access your resources, not customer
data**

- Full Access to Stack operations for own Stacks & resources (in shared accounts)
- (Almost) full access to resources
- Customer data access restricted
- Strong audited utility to access customer data for debugging

Q & A

05

Personio



Personio

The People Operating System