
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

COM301

Unleash developer productivity with infrastructure from code

Jeremy Daly (he/him)

CEO & Founder
Ampt Web Services, Inc.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What if our approach to cloud development is fundamentally wrong?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Jeremy Daly

- CEO and founder of Ampt (formerly Serverless Cloud)
- Consult with companies building in the cloud
- 25-year veteran of technology startups
- Started working with AWS in 2009 and started using AWS Lambda in 2015
- Blogger (jeremydaly.com), OSS contributor, speaker
- Publish the **Off-by-none** serverless newsletter
- Founder and co-host of the **Serverless Chats** podcast



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

I am a developer that thinks building cloud apps is too hard

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

How building applications has changed

The benefits and limitations of Infrastructure **as** Code

The Infrastructure **from** Code approach

The evolving ecosystem of IfC and adaptive architecture

Key takeaways and further reading

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Single server versus multi-service apps

- We don’t build single server applications anymore
- We build distributed systems by composing multiple cloud services
- Multi-service applications introduce all kinds of cross-service complexity
  - AWS Identity and Access Management (IAM) permissions
  - Timeouts
  - State management
  - Network latency
  - Failover/retries



Dr. Werner Vogels, VP and CTO at Amazon.com

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cloud is the new computer

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cloud native versus native cloud apps

The CNCF defines **“cloud native”** as “technologies that empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.”

Allan Vermeulen

22 years at Amazon.com

One of the original creators of Amazon S3

- AWS built these cloud services to achieve things that could **never** be solved at a data center
- The types of apps that can be built on a public cloud **cannot** be ported back because they lack the primitives
- The cloud introduces new levels of scale and capabilities that on-premises environments are **not capable** of providing



This incredible power is available to you **right now**, on-demand, and you generally only pay for what you use!

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“Cloud software is no longer written. It is composed.”

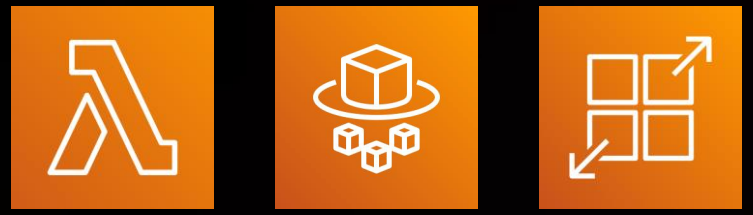
André Eriksson

Founder at Encore

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Core primitives

Compute



Database



Storage



Application integration



Content delivery and analytics



Plus over 200 more

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

We need a way to compose and configure these services

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Infrastructure as Code

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why Infrastructure as Code?

- Configurations became complex
- Avoid drift from human intervention
- Reproducibility of environments
- Improved transparency and collaboration

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Declarative approach

template.yml — scratch

template.yml

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: Amazon API Gateway REST API with IAM authorizer

Resources:
  # IAM user with sufficient privileges to call endpoint
  AuthorizedUser:
    Type: AWS::IAM::User
    Properties:
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: execute-api
                Resource: arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:*/Prod/*
              - Effect: Allow
                Action: lambda:InvokeFunction
                Resource: !GetAtt AppFunction.Arn

  # IAM user missing required privilege to invoke Lambda function
  UnauthorizedUser:
    Type: AWS::IAM::User
    Properties:
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: execute-api:Invoke
                Resource: !Sub "arn:aws:execute-api:${AWS::Region}:${AWS::AccountId}:*/*/*"

  # REST API using IAM as Authorizer
  AppApi:
    Type: AWS::Serverless::Api
    Properties:
```

Custom DSLs

HashiCorp Terraform

serverless framework

AWS SAM

Architect

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Imperative approach

- DSLs started to become more complex
- SLS plugins let you generate declarative code
  - First sign that “declarative wasn’t enough”
  - Shift from config/custom languages to familiar languages (output was YAML, but let you jump into JS)
- New solutions added an imperative syntax to make it easier to programmatically generate resources



AWS Cloud Development Kit



Pulumi

But this still happens once, prior to deployment

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

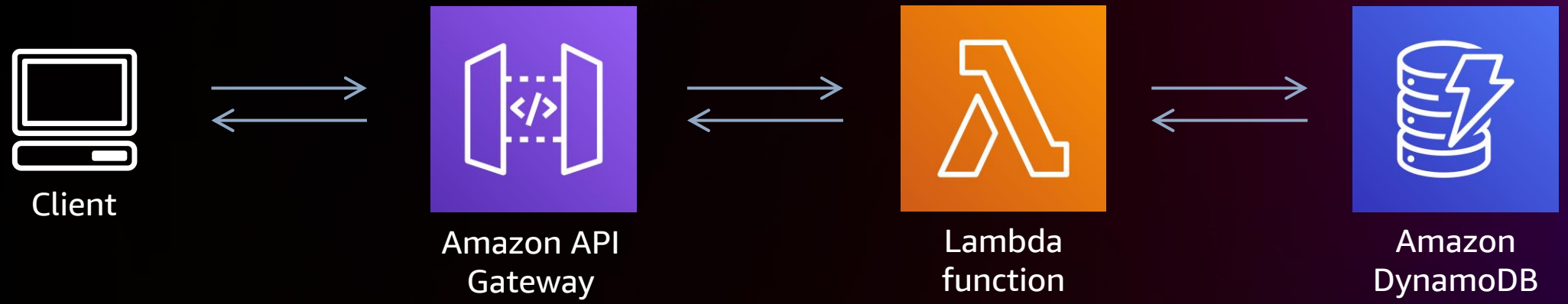
Limitations of Infrastructure as Code

- Duplication and drag as facets are expressed in both code and configuration
- Impedance mismatch between the declarative nature of IaC and the dynamic nature of systems at runtime
- Control-plane specific instructions create static architecture
- Configuration over code (bifurcation of business logic)
- Code is coupled to your architecture



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building a simple web service



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



AWS CDK

```typescript
import * as cdk from '@aws-cdk/core';
import lambda = require('@aws-cdk/aws-lambda');
import dynamodb = require('@aws-cdk/aws-dynamodb');
import apigw = require('@aws-cdk/aws-apigatewayv2');
import integrations = require('@aws-cdk/aws-apigatewayv2-integrations');

export class TheSimpleWebserviceStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    //DynamoDB Table
    const table = new dynamodb.Table(this, 'Hits', {
      partitionKey: { name: 'path', type: dynamodb.AttributeType.STRING }
    });

    // defines an AWS Lambda resource
    const dynamoLambda = new lambda.Function(this, 'DynamoLambdaHandler', {
      runtime: lambda.Runtime.NODEJS_12_X,      // execution environment
      code: lambda.Code.fromAsset('lambda-fns'),  // code loaded from the "lambda" directory
      handler: 'lambda.handler',                // file is "lambda", function is "handler"
      environment: {
        HITS_TABLE_NAME: table.tableName
      }
    });

    // grant the lambda role read/write permissions to our table
    table.grantReadWriteData(dynamoLambda);

    // defines an API Gateway Http API resource backed by our "dynamoLambda" function.
    let api = new apigw.HttpApi(this, 'Endpoint', {
      defaultIntegration: new integrations.LambdaProxyIntegration({
        handler: dynamoLambda
      })
    });

    new cdk.CfnOutput(this, 'HTTP API Url', {
      value: api.url ?? 'Something went wrong with the deploy'
    });
  }
}
```

serverless framework

```yaml
service: simple-web-service
provider:
  name: aws
  runtime: nodejs12.x
  stage: ${opt:stage,'dev'}
  region: us-east-1
  stackName: ${self:service}-${self:provider.stage}
  stackTags:
    SERVICE: ${self:service}
  httpApi:
    payload: '2.0'
    cors: true

functions:
  GetUser:
    handler: index.handler
    memorySize: 1024
    timeout: 6
    tracing: Active
    environment:
      TABLE_NAME: !Ref Users
      TABLE_ARN: !GetAtt Users.Arn
    iamRoleStatementsName: ${self:service}-${self:provider.stage}-getuser-role
    iamRoleStatements:
      - Effect: Allow
        Action:
          - dynamodb:getItem
          - dynamodb:putItem
          - dynamodb:updateItem
          - dynamodb:deleteItem
        Resource:
          - !Join [ '/', [ !GetAtt Users.Arn, '*' ] ]
          - !GetAtt Users.Arn
      - Effect: Allow
        Action:
          - xray:PutTraceSegments
          - xray:PutTelemetryRecords
        Resource: "*"
    events:
      - httpApi:
          path: /user/{id}
          method: get

plugins:
  - serverless-iam-roles-per-function

resources:
  Resources:
    Users:
      Type: AWS::DynamoDB::Table
      Properties:
        AttributeDefinitions:
          - AttributeName: id
            AttributeType: S
        BillingMode: PAY_PER_REQUEST
        KeySchema:
          - AttributeName: id
            KeyType: HASH
        StreamSpecification:
          StreamViewType: NEW_AND_OLD_IMAGES
```



AWS SAM

```yaml
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  HttpApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      DefinitionBody:
        openapi: '3.0'
        info:
          title: !Sub ${AWS::StackName}-HttpApi
          version: '1.0'
        paths:
          /user/{id}:
            get:
              responses: {}
      FailOnWarnings: true
  GetUser:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: !Sub ${AWS::StackName}-GetUser
      Description: !Sub
        - Stack ${StackTagName} Environment ${EnvironmentTagName} Function ${ResourceName}
        - ResourceName: GetUser
      CodeUri: src/getUser
      Handler: index.handler
      Runtime: nodejs12.x
      MemorySize: 1024
      Timeout: 6
      Tracing: Active
      Policies:
        - AWSXrayWriteOnlyAccess
        - DynamoDBCrudPolicy:
            TableName: !Ref Users
      Events:
        HttpApiGETuserid:
          Type: HttpApi
          Properties:
            Path: /user/{id}
            Method: GET
            ApiId: !Ref HttpApi
            PayloadFormatVersion: '2.0'
            TimeoutInMillis: 6000
      Environment:
        Variables:
          TABLE_NAME: !Ref Users
          TABLE_ARN: !GetAtt Users.Arn
  Users:
    Type: AWS::DynamoDB::Table
    Properties:
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: S
      BillingMode: PAY_PER_REQUEST
      KeySchema:
        - AttributeName: id
          KeyType: HASH
      StreamSpecification:
        StreamViewType: NEW_AND_OLD_IMAGES
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Write some application code

```
import express from "express";
import cors from "cors";
import { DynamoDBClient } from "@aws-sdk/client-dynamodb"

const app = express();
const ddbClient = new DynamoDBClient();
app.use(cors());

app.get("/user/:id", async (req, res) => {
  const params = {
    TableName: "Users", Key: { pk: { N: req.params.id } }
  };
  const user = await ddbClient.send(new GetItemCommand(params));
  res.send({ user })
})
```

Looks like you're creating an API with a GET route

And you need GetItem permissions to a DynamoDB table named "Users"

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Interpretive approach

- Resources are inferred by the application code
- Permissions are assumed based on service interactions
- Behavioral semantics are declared by event handlers
- Inter- and intra-service communications are automatically mapped
- Code is decoupled from the underlying infrastructure



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

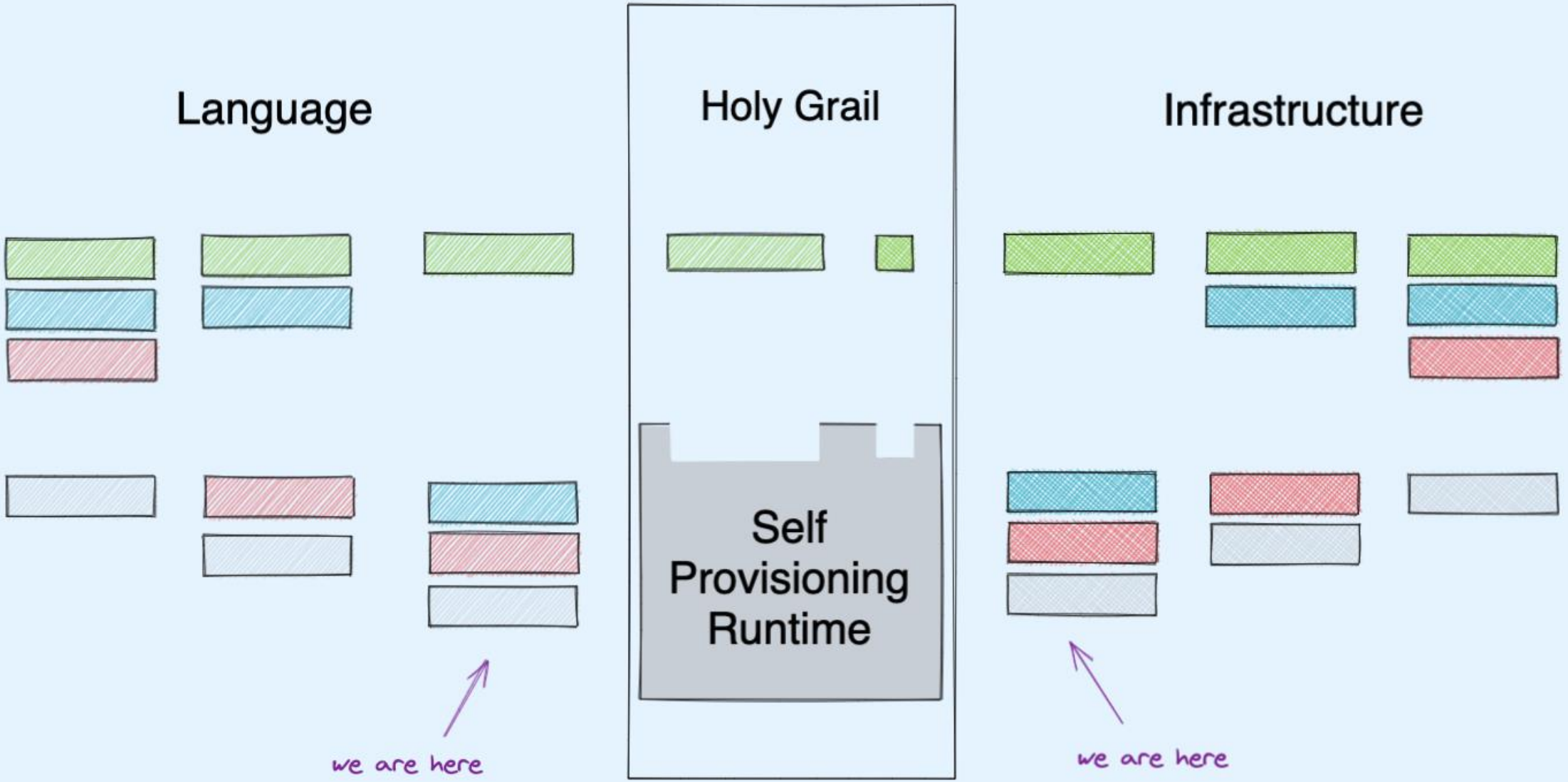
"Advancements in two fields – programming languages and cloud infrastructure – will converge in a single paradigm: where all resources required by a program will be **automatically provisioned**, and **optimized**, **by the environment that runs it.**"

Shawn "swyx" Wang

Legend of DX, Formerly Temporal/Netlify

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Image by Shawn @swyx Wang. https://swyx.io/self-provisioning-runtime

Infrastructure from Code

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Instead of writing low-level, control-plane specific instructions, Infrastructure from Code infers requirements from application logic and provisions the optimal cloud infrastructure

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How does this actually work?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

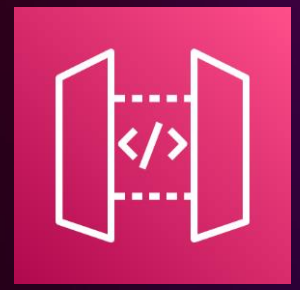
APIs and CDNs

```
// Create a GET route
api.get('/user/:id', (req, res) => {
  // get a user by id
});

// Create a POST route for /users
api.post('/users', (req, res) => {
  // add a new user
});

// Add a timeout
api.delete('/user/:id', { timeout: 2000 }, (req, res) => {
  // delete a user
});
```







aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data store and change data capture

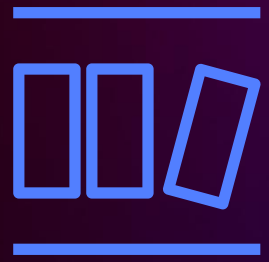
```
// Set user #1
data.set("user:1", {
  name: "Jeremy Daly", twitter: "@jeremy_daly"
});

// Get user #1
const user = data.get("user:1");

// Get all users
const users = data.get("user:*");

// Listen for data changes
data.on(["created", "updated"], (event) => {
// an item has been created or updated
});
```







aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Crontabs and scheduled tasks

```
schedule.every("1 hour", () => {
  // This code block will run every hour!
  console.log("I run every hour!");
});

schedule.cron("0 0 ? * TUE *", () => {
  // This code block will run at midnight on Tuesdays!
  console.log("I run on Tuesdays!");
});

schedule.every("12 hours", { timeout: 1200000 }  () => {
  // This will run every 12 hours and timeout after 20 mins!
  console.log("I run every 12 hours!");
});
```







aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Object storage

```
storage.write("/jeremy.jpg", binaryData,
  { metadata: { isPhoto: true } }
);

const myFile = storage.read("/jeremy.jpg");

storage.copy("/jeremy.jpg", "jeremy-copy.jpg");

const url = storage.getDownloadUrl("/jeremy-copy.jpg");

storage.on("write:user-uploads/*", async (event) => {
  // event = { path: user-uploads/picture.jpg, name: 'write' }
});
```







aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Events, queues, and pub/sub

```
// Publish an event
events.publish("user.joined", {
  user_id: "123456789",
  email: "jeremy@getampt.com"
});

// Publish an event and delay for 1 day
await events.publish("user.joined", { after: "1 day" }, {
  user_id: "123456789",
  email: "jeremy@getampt.com"
});

events.on("user.joined", { batch: 10, ordered: true }, async (event) => {
  // send welcome emails
});
```



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Infrastructure from Code is just the interface

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

When you understand the code . . .



self-provisioning



self-instrumenting



self-documenting



self-testing



self-auditing



self-managing



self-optimizing



self-upgrading

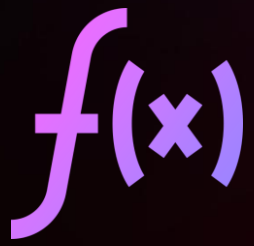
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Portability across technology



Managed services

For critical workloads like event buses, queues, databases, and orchestration; the best of breed provides ultimate reliability



Serverless compute

For low-volume, highly variable, or event-driven workloads; serverless functions instantly go from zero to scale



At the edge

For super low-latency rendering, routing, caching, and header manipulation; edge functions put your workloads close to the user



Orchestrated containers

For high-volume, sustained workloads; containers give apps the perfect combination of cost optimization and performance

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Adaptive architecture

- Auto scaling, automated failover, built-in resiliency
- Self-indexing databases
- Reactive telemetry/observability data
  - Memory
  - Timeouts
  - Latency
  - Service dependencies
  - Predictive scaling



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“But I need more control!”

Everyone sitting in this room 

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“You never had control – that’s the illusion!”

Dr. Ellie Sattler

Paleobotanist Consultant, *Jurassic Park*

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Not just “someone else’s computer”

- It’s a complex, globally distributed system that requires constant maintenance and optimizations that you don’t know how to make or have the ability to make

And that’s a good thing!

- AWS has lots of “knobs” that you can turn in their managed services and compute solutions
- Most performance tuning beyond code optimizations and architecture choices are out of your control



Pro tip: If you think your application has reached the limits of a cloud service, reach out to the service team, and they’ll be more than happy to work with you to make the service better

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

We’re closer than you might think

















aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why people believe in this approach

klotho

Decoupling code from infrastructure enables “adaptive architectures”; code is the spec and doesn’t change, but the infrastructure can

shuttle

We are an orchestrator of AWS resources using the Rust API – IaC produces outputs that your code needs to wire together; Rust fills in the gap

encore

IfC enables tools that can understand the structure of your app, and reason about how it interacts with infrastructure and other services

Modal

Everything needs to run in the cloud, not on your laptop – ephemeral environments with fast feedback loops

nitric

It’s about portability; we want to help developers delay making decisions until they know what resources their app needs

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

An operating system for the cloud



- **Only** write application code
- No YAML or configuration files
- Focus on **use cases** and **outcomes** versus primitives
- Use familiar patterns to define business logic
- Automatically deploys/manages infrastructure to support the app
- Provide simple workflows for development, testing, and deployments
- Integrates and orchestrates third-party apps (SaaS, APIs, other cloud services)
- Infrastructure is optimized over time based on runtime behavior

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Key takeaways

- We build modern cloud apps by composing multiple cloud services
- Building multi-service, native cloud applications is really hard
- Infrastructure **as** Code tightly couples your app code to static architecture
- Higher-level abstractions can still provide control while reducing redundancy
- Infrastructure **from** Code can:
  - Accelerate development
  - Decouple business logic from underlying infrastructure
  - Understand the structure of your application
  - Enable platforms to optimize over time

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Further reading

- **Infrastructure from Code** by the Ampt team
  https://infrastructurefromcode.com
- **The Self-Provisioning Runtime** by Shawn “swyx” Wang
  https://swyx.io/self-provisioning-runtime
- **What Does The Future Hold For Serverless?** by Allen Helton
  https://www.readysetcloud.io/blog/allen.helton/what-does-the-future-hold-for-serverless/
- **Infrastructure-from-Code vs Infrastructure-as-Code** by Paul Swail
  https://serverlessfirst.com/emails/infrastructure-from-code-vs-infrastructure-as-code/
- **Overcoming IaC obstacles: It’s time for Infrastructure from Code** by Asher Sterkin
  https://www.idgconnect.com/article/3647074/overcoming-infrastructure-as-code-obstacles-its-time-for-infrastructure-from-code.html

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Jeremy Daly
@jeremy_daly
jeremydaly.com



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.