

Scheduled Tasks Without a Server

James Hall, Director, Parallax
@MrRio

Who we are

We help the world's top companies and most ambitious startups build brilliant digital products, services and applications.

parallax



M P & SILVA



@parallax

Serverless Applications on AWS

If you're following along... run this now

PARALLAX

```
npm install -g serverless@alpha
```

parallax

Cron

@parallax

What is cron?

Cron is a task scheduler that can run as often as every minute. It first shipped with Version 7 Unix in 1979.

What is it good for?

In web applications, examples include:

- Sending out subscription expiry emails
- Checking uptime
- Archiving files on S3

parallax

crontab

@parallax

The Basics

```
crontab -e
```

```
* * * * * node /var/www/my/script.js
```

PARALLAX

| | | | |

| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)

| | | ----- Month (1 - 12)

| | ----- Day of month (1 - 31)

| ----- Hour (0 - 23)

----- Minute (0 - 59)

What's wrong with cron?

Modern Applications

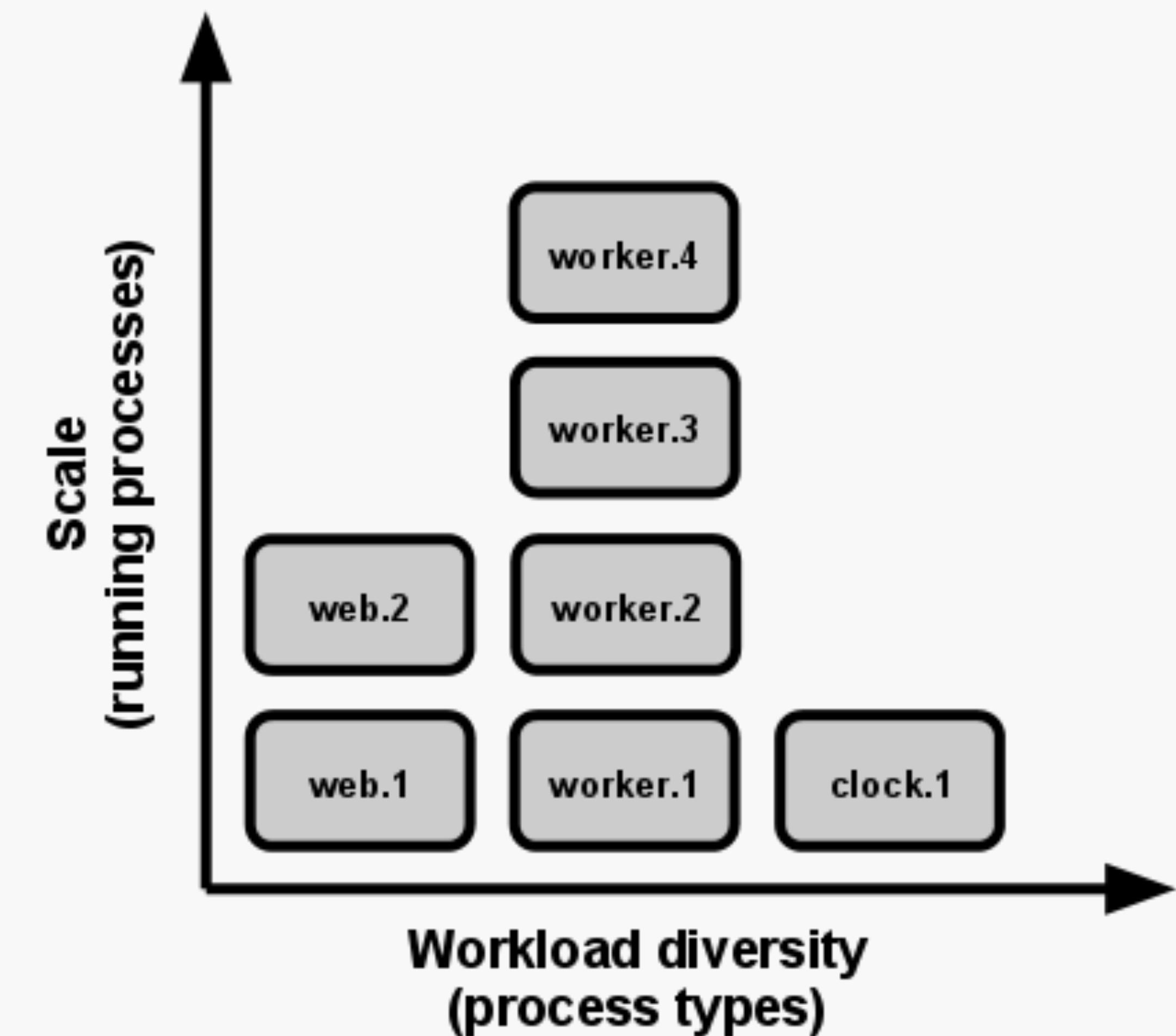
Modern web applications follow the Twelve Factors
(See: <http://12factor.net/>)

PARALLAX

Rule 8 is **Concurrency**

Rule 9 is **Disposability**

Crontabs don't work so well here



Serverless Framework

@parallax

Serverless Applications on AWS

Installing

```
npm install -g serverless@alpha
```

Setting AWS Credentials

Network

Elastic File System

Fully Managed File System for EC2

Glacier

Archive Storage in the Cloud

Snowball

Large Scale Data Transport

Storage Gateway

Hybrid Storage Integration

Database

RDS

Managed Relational Database Service

DynamoDB

Managed NoSQL Database

ElastiCache

In-Memory Cache

Redshift

Fast, Simple, Cost-Effective Data Warehousing

DMS



CloudWatch

Track User Activity and API Usage



Config

Track Resource Inventory and Changes



OpsWorks

Automate Operations with Chef



Service Catalog

Create and Use Standardized Products



Trusted Advisor

Optimize Performance and Security

Security & Identity



Identity & Access Management

Manage User Access and Encryption Keys



Directory Service

Host and Manage Active Directory



Inspector

Find and Fix Cloud Configuration Errors



Test Android, iOS, and web Apps on Real Devices in the Cloud



Mobile Analytics

Collect, View and Export App Analytics



SNS

Push Notification Service

Application Services



API Gateway

Build, Deploy and Manage APIs



AppStream

Low Latency Application Streaming



CloudSearch

Managed Search Service



Elastic Transcoder

Easy-to-Use Scalable Media Transcoding



SES

Email Sending and Receiving Service



SQS

View your resources on the go with our AWS Console mobile app, available from [Amazon Appstore](#), [Google Play](#), [iTunes](#).

AWS Marketplace

Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements



Explore the next generation of AWS cloud capabilities. [See what's new](#)

Service Health



All services operating normally.

Updated: Jul 21 2016 17:00:00 GMT+0100

[Service Health Dashboard](#)

Setting AWS Credentials

The screenshot shows the AWS IAM User Details page for a user named 'uefa'. The URL in the browser is <https://console.aws.amazon.com/iam/home?region=us-east-1#users/uefa>. The top navigation bar includes links for Dashboard, AWS Services, Edit, James Hall, and Glossary. On the left, a sidebar lists options like Details, Groups, Users (which is selected), Roles, Policies, Identity Providers, Account Settings, Credential Report, and Encryption Keys. The main content area displays the user's ARN, password status, groups, path, and creation time. Below this, tabs for Groups, Permissions, Security Credentials (selected), and Access Advisor are shown. The Security Credentials tab contains a section for Access Keys, explaining their use and best practices, with a 'Create Access Key' button.

Dashboard

AWS Services Edit James Hall Glossary

IAM > Users > uefa

Search IAM

Details

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

User ARN: arn:aws:iam::190768418884:user/uefa

Has Password: Yes

Groups (for this user): 0

Path: /

Creation Time: 2015-12-15 15:44 UTC+0100

Groups Permissions **Security Credentials** Access Advisor

Access Keys

Use access keys to make secure REST or Query protocol requests to any AWS service API. For your protection, you should never share your access keys with anyone. In addition, industry best practice recommends frequent key rotation. [Learn more about Access Keys](#)

Create Access Key

Setting AWS Credentials

The screenshot shows the AWS IAM service interface. On the left, a sidebar lists navigation options: Dashboard, Search IAM, Details, Groups, **Users**, Roles, Policies, Identity Providers, Account Settings, Credential Report, and Encryption Keys. The 'Users' option is currently selected. In the main content area, the path 'IAM > Users > uefa' is displayed, and the 'Summary' section is open. The summary details include:

- User ARN: arn:aws:iam::190768418884:user/uefa
- Has Password: Yes
- Groups (for this user): 0
- Path: /
- Creation Time: 2015-12-15 15:44 UTC+0100

A modal window titled 'Create Access Key' is overlaid on the page. It contains the following message:
✓ Your access key has been created successfully.
This is the last time these User security credentials will be available for download.
You can manage and recreate these credentials any time.
[▼ Hide User Security Credentials](#)

The modal also displays the generated credentials:
Access Key ID: ACCESS KEY
Secret Access Key: SECRET ACCESS KEY

At the bottom of the modal are two buttons: 'Close' and 'Download Credentials'.

Setting AWS Credentials

~/.aws/credentials

PARALLAX

```
[default]
aws_access_key=THISISYOURACCESSKEY
aws_secret_access_key=ANDYOURSECRET1
```

Making a function

PARALLAX

```
serverless create \  
  --name cron \  
  --provider aws \  
  --template aws-nodejs
```



cron

- node_modules
- handler.js
- package.json
- serverless.env.yaml
- serverless.yaml

handler.js

package.json

serverless.yaml

serverless.env.yaml

```
1 service: aws-nodejs
2 provider: aws
3 runtime: nodejs4.3
4
5 functions:
6   cron:
7     handler: handler.run
8     events:
9       - schedule: rate(1 minute)
10
```



cron

node_modules

handler.js

package.json

serverless.env.yaml

serverless.yaml

handler.js

serverless.yaml

serverless.env.yaml

```
1  'use strict'  
2  
3  var request = require('request')  
4  // Your first function handler  
5  module.exports.run = (event, context, cb) => {  
6      request.post(  
7          'http://requestb.in/yq78y3yq',  
8          { form: { timestamp: new Date() } },  
9          function (error, response, body) {  
10              if (!error && response.statusCode === 200) {  
11                  cb(null, { message: 'Success!', event })  
12              }  
13          }  
14      )  
15  }
```



cron

- > .git
- > node_modules
- .gitignore
- handler.js
- package.json
- serverless.env.yaml
- serverless.yaml

handler.js

package.json

serverless.yaml

```
1  {
2    "name": "cron",
3    "version": "0.0.1",
4    "description": "A simple cron example",
5    "author": "James Hall - Parallax Agency Ltd",
6    "license": "MIT",
7    "dependencies": {
8      "request": "^2.73.0"
9    }
10 }
11 }
```

[Jamess-MacBook-Pro:cron jameshall\$ serverless create --name cron --provider aws --template aws-nodejs
Serverless: Creating new Serverless service...



Serverless: Successfully created service in the current directory
Serverless: with template: "aws-nodejs"

[Jamess-MacBook-Pro:cron jameshall\$ sls deploy

Serverless: Creating Stack...
Serverless: Checking stack creation progress...
Serverless: Stack successfully created.
Serverless: Zipping service...
Serverless: Uploading .zip file to S3...
Serverless: Updating Stack...
Serverless: Checking stack update progress...
Serverless: Deployment successful!

 <http://requestb.in/yg78y3yg>

<http://requestb.in>
POST /yg78y3yg

</> application/x-www-form-urlencoded
 38 bytes

9s ago 
From 52.201.231.235,
162.158.76.143

FORM/POST PARAMETERS

timestamp: 2016-07-21T14:07:32.579Z

HEADERS

Cf-Ray: 2c5f3844b31c0f39-IAD
Content-Type: application/x-www-form-urlencoded
Connection: close
Total-Route-Time: 0
Accept-Encoding: gzip
Cf-Visitor: {"scheme": "http"}
X-Request-Id: 9dd9264a-8093-435c-a72e-ac6c9c45cca5
Via: 1.1 vegur
Content-Length: 38
Connect-Time: 1
Cf-Lpcountry: US
Host: requestb.in
Cf-Connecting-Ip: 52.201.231.235

RAW BODY

timestamp=2016-07-21T14%3A07%3A32.579Z

<http://requestb.in>
POST /yg78y3yg

</> application/x-www-form-urlencoded
 38 bytes

1m ago 
From 52.201.231.235,
173.245.56.95

parallax

Awesome. How
does that work?

@parallax

https://console.aws.amazon.com/cloudwatch/home?region=us-east-1#rules:

AWS Services Edit James Hall N. Virginia Support

CloudWatch Dashboards Alarms ALARM INSUFFICIENT OK Billing Events Rules Logs Metrics Selected Metrics Billing CloudFront DynamoDB Events Lambda Logs S3

Rules

Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules.

Create rule Actions

Status All Name

« < Viewing 1 to 1 of 1 Rules > »

	Status	Name	Description
<input type="radio"/>		aws-nodejs-dev-helloScheduleEvent0-3MPISIX9XG40	



AWS

Services

Edit

James Hall

N. Virginia

Support

CloudWatch
Dashboards

Alarms

ALARM

0

INSUFFICIENT

0

OK

0

Billing

Events

Rules

Logs

Metrics

Selected Metrics

Billing

CloudFront

DynamoDB

Events

Lambda

Logs

S3

Rules > aws-nodejs-dev-helloScheduleEvent0-3MPISIX9XG40

Actions

Summary

ARN arn:aws:events:us-east-1:190768418884:rule/aws-nodejs-dev-helloScheduleEvent0-3MPISIX9XG40

Schedule Fixed rate of 1 minutes

Status Enabled

Description

Role ARN

Monitoring [Show metrics of rules](#)

Targets

Filter: <input type="text"/>			« < Viewing 1 to 1 of 1 Targets > »
Type	Resource name	Input	
Lambda function	aws-nodejs-dev-hello	Matched event	

https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/aws-nodejs-dev-hello?tab=trigg...   ABP   ABCD EFGH IJKL 0      1

AWS Services Edit James Hall N. Virginia Support

Lambda > Functions > aws-nodejs-dev-hello ARN - arn:aws:lambda:us-east-1:190768418884:function:aws-nodejs-dev-hello

Qualifiers Test Actions

Code Configuration Triggers Monitoring ?

 CloudWatch Events - Schedule: [aws-nodejs-dev-helloScheduleEvent0-3MPISIX9XG40](#)  

arn:aws:events:us-east-1:190768418884:rule/aws-nodejs-dev-helloScheduleEvent0-3MPISIX9XG40

Schedule expression: **rate(1 minute)** Description:

 [Add trigger](#)

What about more
complicated scheduling
rules?



cron

handler.js package.json serverless.yaml serverless.env.yaml

```
1 service: aws-nodejs
2 provider: aws
3 runtime: nodejs4.3
4
5 functions:
6   cron:
7     handler: handler.run
8     events:
9       # Run the task on the first day of every month at midnight
10      - schedule: cron(0 0 1 * * *)
```

Slides and code:

<https://github.com/parallax/serverless-cron-example>

Thanks, any questions?

James Hall, Director, Parallax

@parallax
@MrRio