Cloudy Perl

German Perl Workshop 2025

Dave Lambley / DLAMBLEY / dave@lambley.me.uk

In which we use the cloud with CPAN

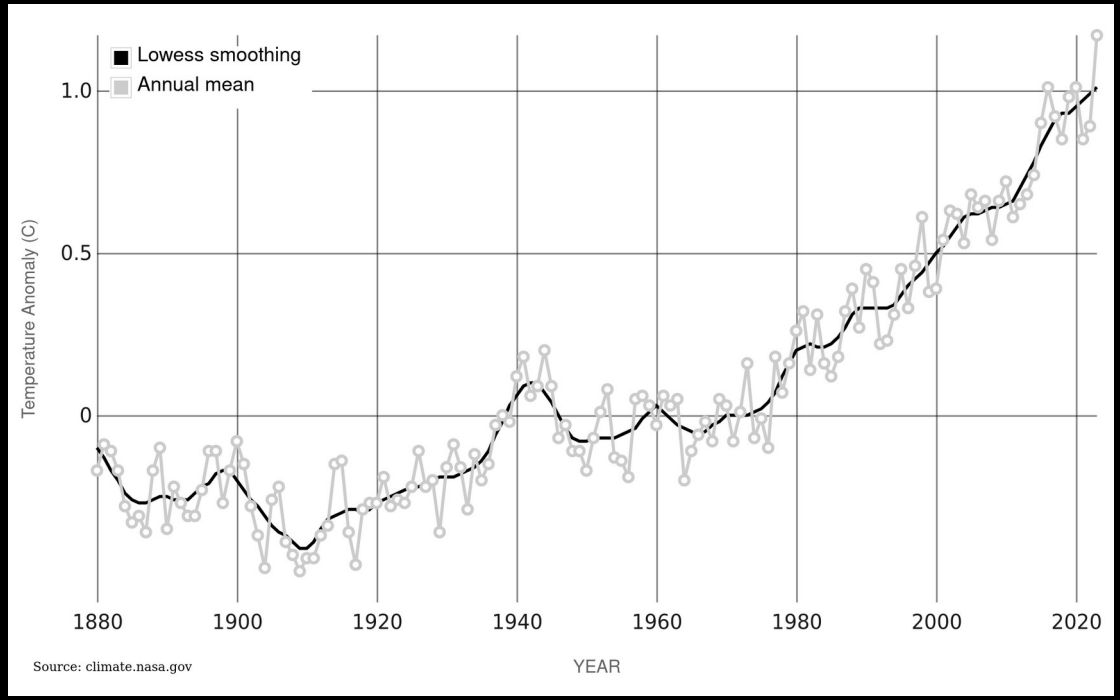
- “Cloud” means Amazon’s.
- Funds private space programme.

Is a private space programme good?

- Jeff proudly burns green hydrogen.

Green hydrogen

- Is a clear liquid.
- Burnt at altitude.
- May produce clouds.



https://climate.nasa.gov/vital-signs/global-temperature/?intent=121

Other clouds do exist

But not in CPAN for;

- Microsoft Azure (Packages available!)
- Google (GCP)
- Scaleway (who did those little ARM boxes years ago)

Playing nicely 1

- Operating the services.
- Making configuration changes.

Playing nicely 2

- Using AWS to run your Perl.

Not retrocomputing

- I have riscose for that.

Where to start

$ zcat 02packages.details.txt.gz | perl -ne'/\/([^\/]+)-[^-]+tar.gz$/; $x=$1; print "$x\n" if $x=~/Amazon|AWS/;' | sort | uniq | wc

118 distributions (119 including Paws.)

“Support” may be important

$ apt search "(aws|amazon).+-perl"

Only 12 in current Debian unstable.

The really important ones are there!

But being simple

system(“/usr/bin/aws”, …)==0 or die;

- Well documented.
- Handles authentication.
- Easily installed (if not already.)

Abstracting away

AWS::CLIWrapper

```
my $res = $aws->ec2(
    'describe-instances' => {
        instance_ids => ['i-XXXXX', 'i-YYYYY'],
    },
);
```

Thank you HIROSE!

AWS::CLIWrapper

- Syntax a little nicer.
- Unpacks output JSON.
- Same advantages of aws-cli.

The specialist modules

(Or, most of the 118.)

Many look abandoned.

The ancient ones

Net::Amazon::S3

- Still relevant!
- API still seems odd.

Net::Amazon::EC2

- EC2 not used like this these days!

The general purpose modules

- Amazon provide JSON API descriptions.
- See also their “botocore” Python project.
- I looked at these once and gave up.

Paws

- Comprehensive.
- Auto-generated code.

eg.,

```
my $result =
$ec2->DescribeInstances(InstanceIds =>
[ 'i-....' ]);
```

Paws for thought

- Not much development happening.

Amazon::API

- Endeavours to be lighter-weight.
- Uses botocore data to generate a wrapper.
- Also a noble achievement!

Going even lighter weight

Invoking API via LWP isn’t too bad! Sharp edges;

- Inconsistent HTTP method.
- Inconsistent error responses.
- Inconsistent HTML form vs. JSON bodies.
- Inconsistent JSON Content-Type (really!)

The easy part

Net::Amazon::Signature::V4

```
my $req = HTTP::Request->new(
    $method,
    "https://$service.$region.amazonaws.com/$query",
    $headers,
    $content,
);

my $signer = Net::Amazon::Signature::V4->new({
    access_key_id => $ENV{AWS_ACCESS_KEY_ID},
    secret        => $ENV{AWS_SECRET_ACCESS_KEY},
    security_token => $ENV{AWS_SESSION_TOKEN},
    endpoint => $region,
    service => $service,
});

return $ua->request($signer->sign($req));
```

Signing requests

- Your request gets a magical HMAC derived from the supplied keys.
- **AWS::SignatureV4** also exists.
- Both packaged by Debian.

You now to get handle responses

- Probably JSON.
- Documentation and examples are good.
- You need this.
- Especially if you care about errors.

Running code

- Lambda
- Fargate / ECS

Lambda

- Runs your routine on demand.
- Can run containers.
- Scaling done by them, paying done by you.

```
while (1) {
    my $resp = $ua_poll->get("http://$ENV{AWS_LAMBDA_RUNTIME_API}/2018-06-01/runtime/invocation/next");

    if ($resp->is_success) {
        my $req_id = $resp->header('Lambda-Runtime-Aws-Request-Id');

        my $req = decode_json($resp->decoded_content);
```

AWS::Lambda

- Wraps this.
- Also allows deployment via Zip file.

Debugging

- “X-Ray”, send debug data via HTTP**.**
- **AWS::XRay**

ECS / Fargate

- ECS provides a Kubernetes-like service.
- But simpler.
- **Fargate** capacity provider frees you from hardware.
- You supply a container image for your **task** or **service**.

Future possibilities

- Terraform CDK integration?
- (OpenTofu)
- Pulumi (Terraform alternative) may be more welcoming.

In conclusion

Perl 5 is in the modern world!

Thank you!

Questions?

dave@lambley.me.uk / https://dave.lambley.me.uk/