



Accounts as a Service

*Why we have 50+ AWS accounts, and
why you should too*

Keith Gregory

AWS Practice Lead, Chariot Solutions

About Chariot Solutions

50-person application-development consulting firm

Created this conference!

Offers training and developer workshops

AWS Consulting Partner



Why we did this

Reduce friction for workshop attendees

They might not have (or be able to use) an existing account

They could create an account, but this takes time

Support project teams



Why you might want to do the same

Allow developers/teams space to experiment

Ensure that they don't harm others

Centralizes control, billing

Good practice for managing/promoting workloads



Approaches we considered

Single account, segregated by region/VPC/naming

Manually create/manage independent accounts

Manually create/manage accounts within organization

AWS Control Tower



Goals

Limit the likelihood of unexpected (big) bills

Identify who is responsible for creating what

Automate all the things!

Setting Up Your Organization

Control Tower

Amazon's enterprise-scale multi-account solution

Account Factory to create child accounts

Predefined “guardrails” based on AWS best practices

Accounts for Application and Audit (CloudTrail) logs

Too heavyweight?

AWS Organizations

Supports a hierarchy of accounts

Individual accounts are assigned to an “organizational unit,” which allows controls to be applied to the entire group

Accounts can be invited or created

Each child account needs a unique email address (“+name” FTW)

You will need to increase your account limit

Default limit is 4 accounts per organization

AWS won't increase limits on new accounts

Organizing Accounts

The screenshot displays the AWS Organizations console in a Mozilla Firefox browser. The top navigation bar includes the AWS logo and links to various services: Services, Resource Groups, IAM, CloudFormation, CloudWatch, EC2, Elastic Container Service, Lambda, and a user profile for 'kgregory @ chariotsolutions'. The main header shows 'AWS Organizations' with tabs for 'Accounts', 'Organize accounts', 'Policies', 'Invitations', and 'Settings'.

The left sidebar features a tree view with the following structure:

- Root
 - Sandboxes
 - WorkshopUsers


The main content area is titled 'Sandboxes' and includes a 'TREE VIEW' toggle and a search filter. It displays the following information:

- Organizational units (0): A button to '+ New organizational unit' and a message: 'No organizational units are present in the current OU'.
- Accounts (12): A grid of 12 accounts, each with a checkbox and a name. The accounts are:
 - developer-sandbox (aws-developer-sandbox@cha...)
 - Developer Sandbox #4 (aws-sandbox-4@chariotsol...)
 - Developer Sandbox #7 (aws-sandbox-7@chariotsol...)
 - Developer Sandbox #2 (aws-sandbox-2@chariotsol...)
 - Developer Sandbox #3 (aws-sandbox-3@chariotsol...)
 - Developer Sandbox #8 (aws-sandbox-8@chariotsol...)
 - Developer Sandbox #6 (aws-sandbox-6@chariotsol...)
 - krimple-sandbox (krimple-sandbox@chariotsol...)
 - Developer Sandbox #5 (aws-sandbox-5@chariotsol...)
 - kgregory-sandbox (kgregory-sandbox@chariotsol...)
 - Fall Marketing Event 2019 (aws-fall-event-2019@chariotsol...)
 - Developer Sandbox #1 (aws-sandbox-1@chariotsol...)

The right-hand panel provides details for the selected 'Sandboxes' organizational unit:

- ARN: arn:aws:organizations::123456789012:ou-123456789012
- ID: ou-123456789012
- Accounts: A link to view all accounts under this OU.
- POLICIES: A section with links to 'Service control policies' and 'Tag policies'.

The bottom of the page features a footer with 'Feedback', 'English (US)', and copyright information: '© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.



Creating a child account

Can only be invoked from organization root account

Each child account must have a unique email address

Also creates OrganizationAccountAccessRole

```
aws organizations create-account \  
--email sandbox+1@example.com
```

User Management

Option 1: Create IAM users in each child account

Easiest for users ... as long as they only have one account

Option 2: Create IAM users in parent account

Users log-in and then assume role; better for multiple accounts

Option 3: AWS Single Sign-On

Uses an existing Active Directory server to manage users

AD User \Rightarrow AD Group \Rightarrow SSO Permission Set \Rightarrow IAM Role



ProTip: use multi-factor authentication

Goal: weak/leaked password != big bill

AWS supports several options

- Virtual TOTP (Google Authenticator, Authy, ...)

- Physical TOTP (Gemalto)

- Hardware key (Yubikey)

Organization Account Access Role

Administrator role in each child account

Created automatically when child is created inside organization

Must be manually created for invited accounts

Default name: OrganizationAccountAccessRole

Damaging or deleting this role is a Bad Idea™

“Guardrails”

Service Control Policies

IAM-like policy that limits what a child account can do

If an SCP prevents an activity, it doesn't matter if the account allows it — root user notwithstanding!

Does not replace IAM: accounts must define own policies

Can be applied to account or organizational unit

Allow versus Deny SCPs

Allow lets you do something, Deny prevents it

Example: Allow EC2

Gets tricky when multiple SCPs apply to an account

Example: Allow-All, Deny EC2

Allow applies only to accounts in specific organizational unit

Deny applies to all descendents, overrides Allow

Don't apply Deny-All at root of organization!



Your First SCP

The organization access role is your way to manage the account without an explicit login; don't let it get damaged!

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ProtectOrgAccess",
      "Effect": "Deny",
      "Action": [
        "iam:DeleteRole",
        "iam:DeleteRolePolicy",
        "iam:DetachRolePolicy",
        "iam:UpdateRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/OrganizationAccountAccessRole",
      ]
    }
  ]
}
```



Limit the regions you allow

You can be more clever, and allow read-only access, but probably not worth it.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictRegion",
      "Effect": "Deny",
      "Action": [ "*" ],
      "Resource": [ "*" ],
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:RequestedRegion": [
            "us-east-1",
            "us-east-2",
            "us-west-1",
            "us-west-2"
          ]
        }
      }
    }
  ]
}
```

Limit expensive resources

You'll also need to do this for RDS, Redshift, Elasticsearch, and others.

Some services (eg, Private Certificate Authorities) should be disabled entirely.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LimitEC2InstanceType",
      "Effect": "Deny",
      "Action": [
        "ec2:RunInstances"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/*"
      ],
      "Condition": {
        "StringNotLike": {
          "ec2:InstanceType": [
            "t2.*",
            "t3.*",
            "m5.large",
            "m5d.large"
          ]
        }
      }
    }
  ]
}
```

Turning Your Users Loose

Have “The Talk” about security

Review AWS Shared Responsibility Model

Example: security groups

Long random passwords + multi-factor auth FTW

Access keys

“Keep them secret, keep them safe!”

Assumed roles (and SSO) make them less vulnerable

Don’t fire up random AWS Marketplace AMIs

Tag Resources

Start with a simple strategy

CreatedBy: username of person who created resource

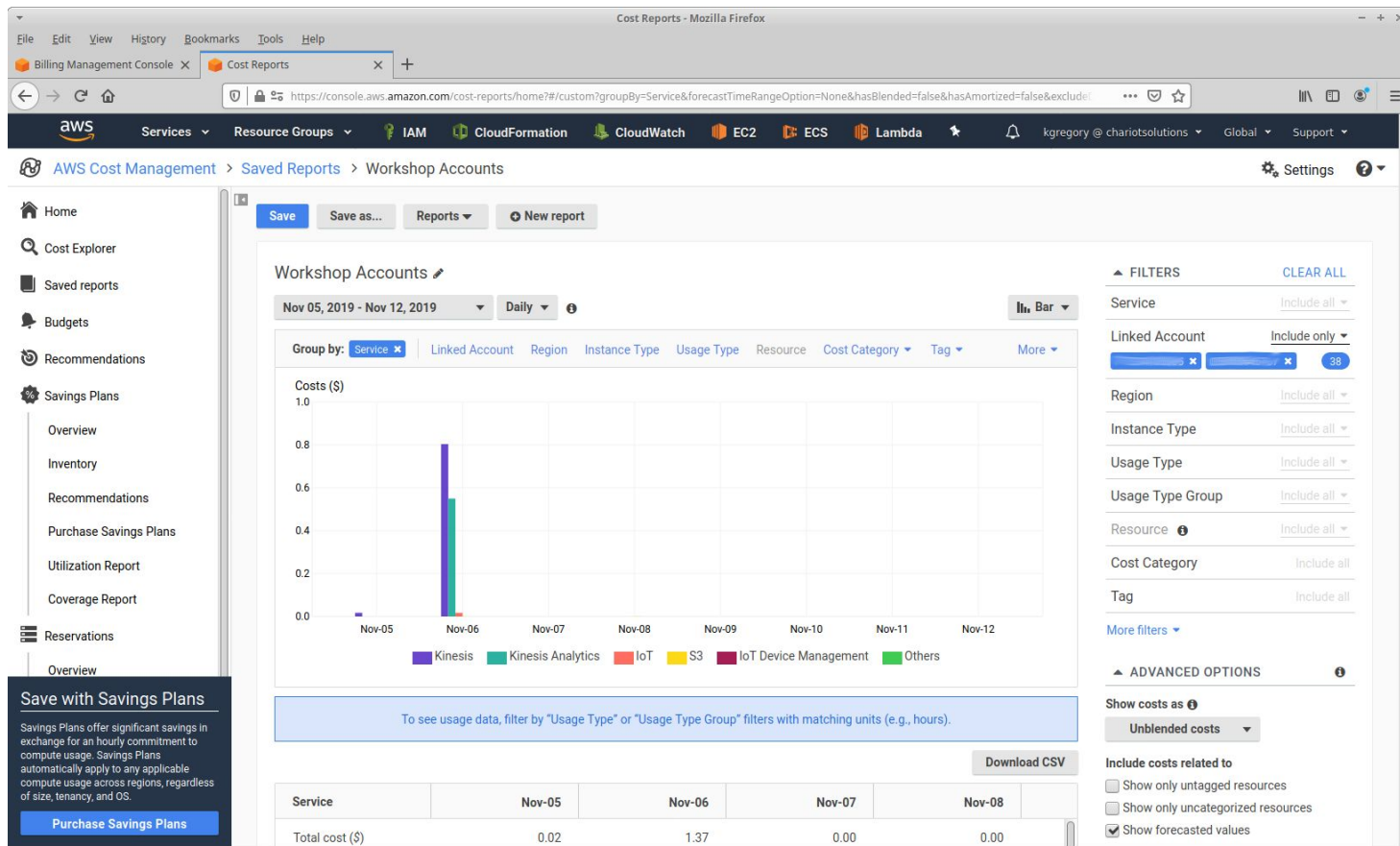
DeleteAfter: how long they expect to need it

Implement a Lambda to look for missing tags

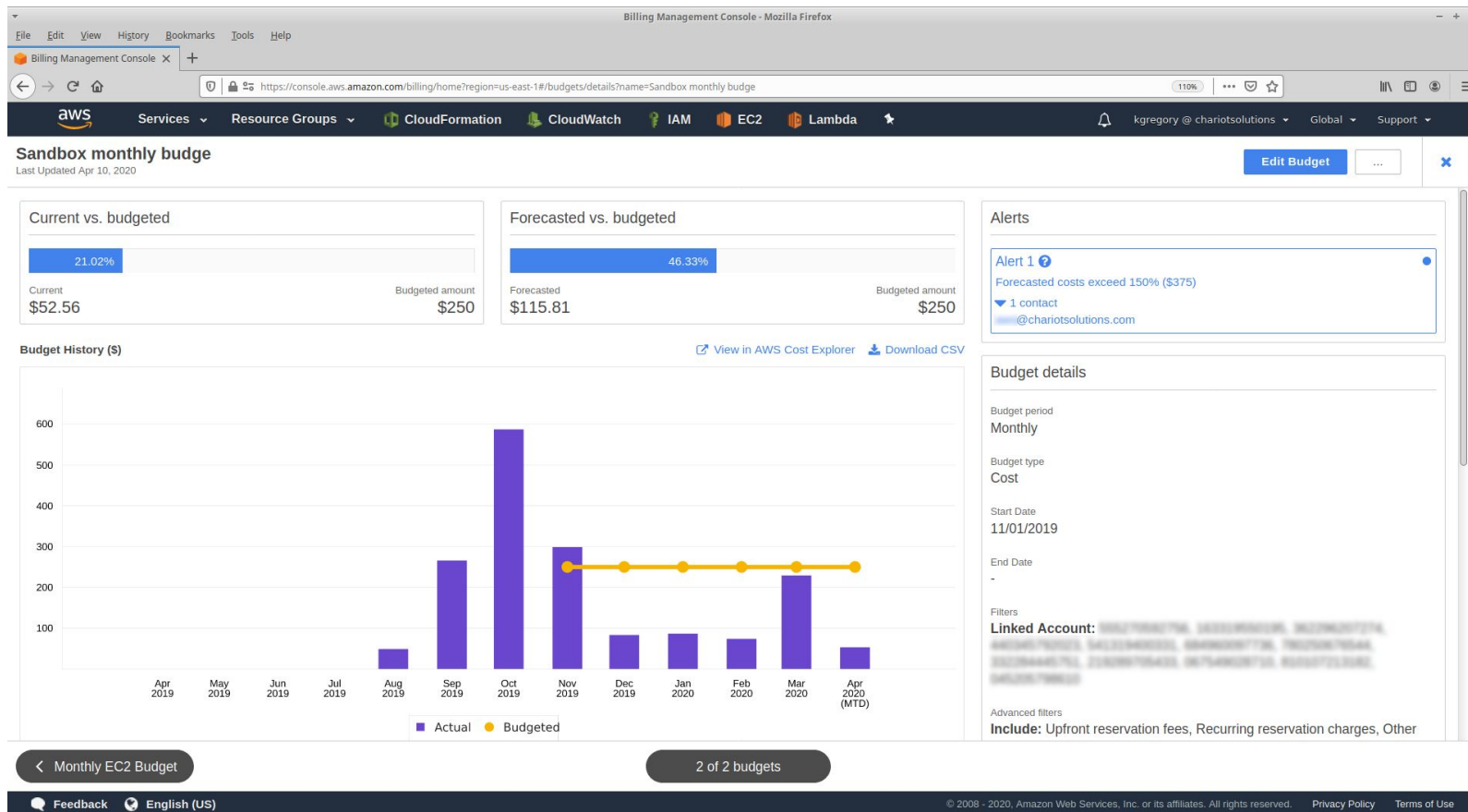
Send an SNS notification, or just delete the resource

This is actually a complex task: multi-service, multi-region,
multi-account

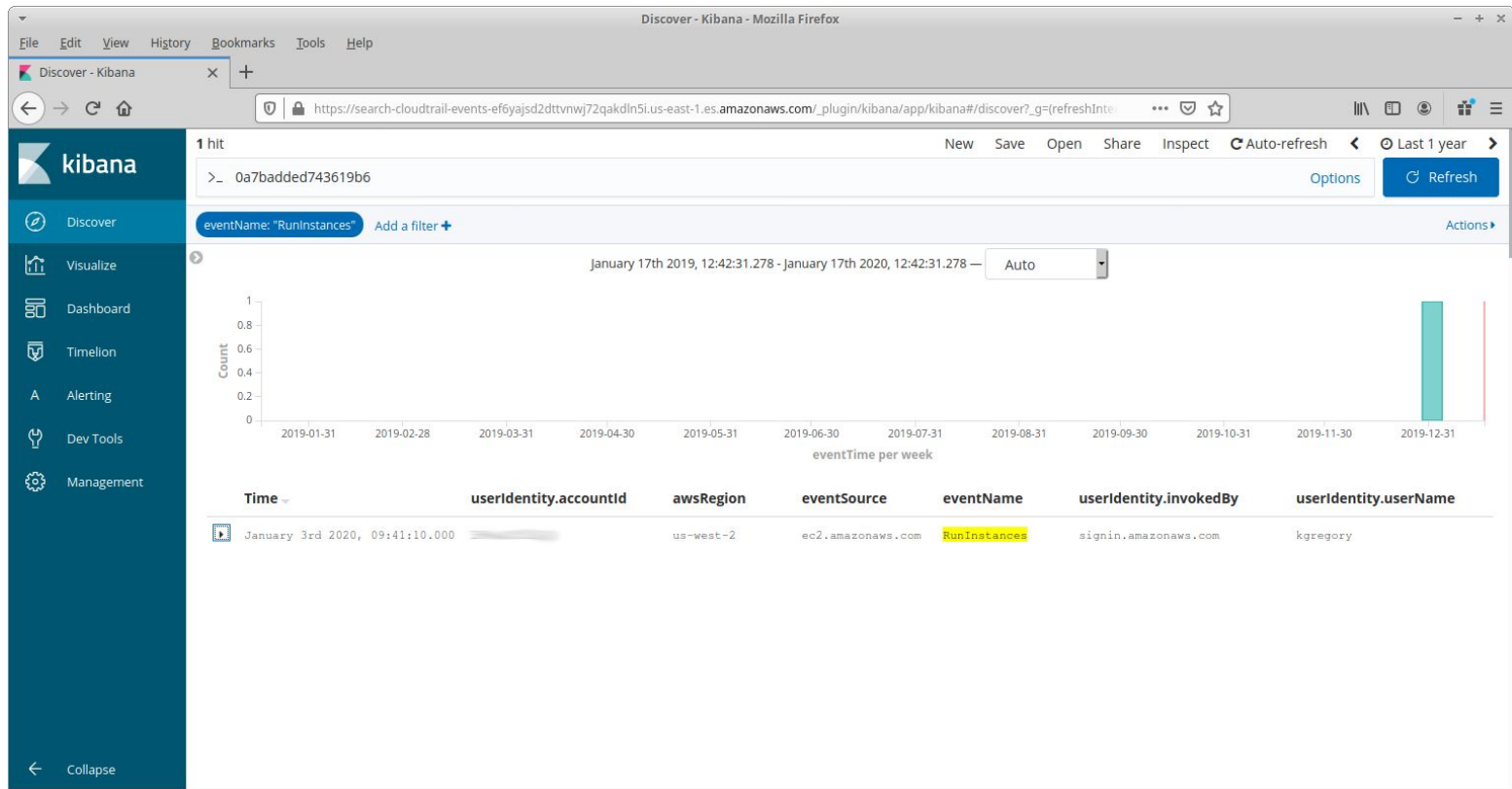
Cost Explorer is your new BFF



Along with AWS Budgets



And CloudTrail + Search Engine



If You're Running a Workshop

Getting Ready

Identify the services used by the workshop and craft an Explicit-Allow SCP that allows all of them

Test-run the workshop to ensure you've caught them all!

Create a “workshop” IAM user for each account, with login profile and randomly-generated password

Attach a “deny all” SCP until the day of the workshop

Wrapping-up

Attach an SCP that prevents creating new resources

Run a script to delete all resources for the accounts

First thing to delete: user access keys, login profiles, non-standard users

There will be some cases where manual intervention is needed

Attach a “deny all” SCP until next time



ProTip: Excel is Your Friend

Keep track of all of your accounts in one sheet

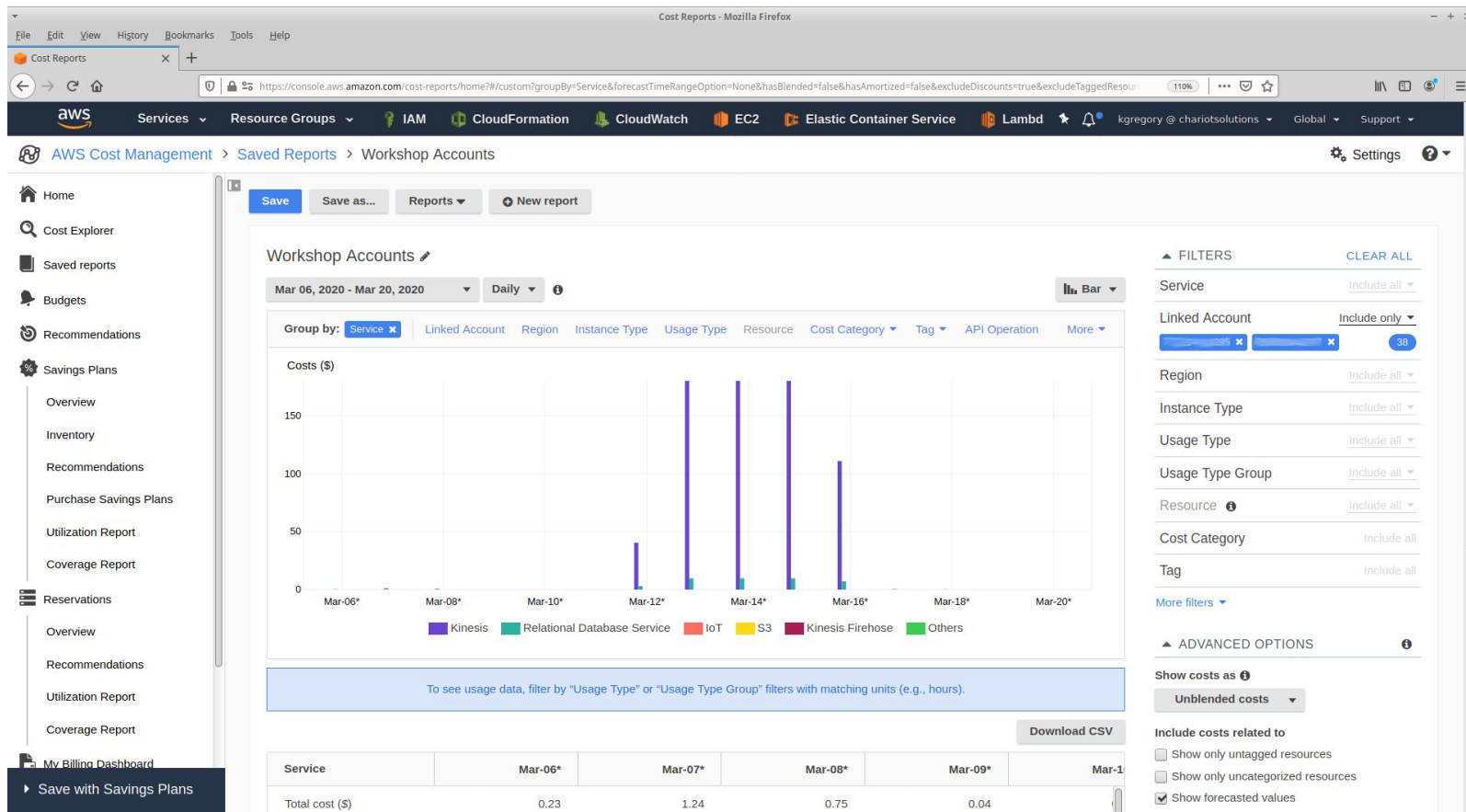
Randomly generate passwords

```
=TRUNC(10000000000*RAND())
```

No, this isn't secure, but it's a limited-duration vulnerability

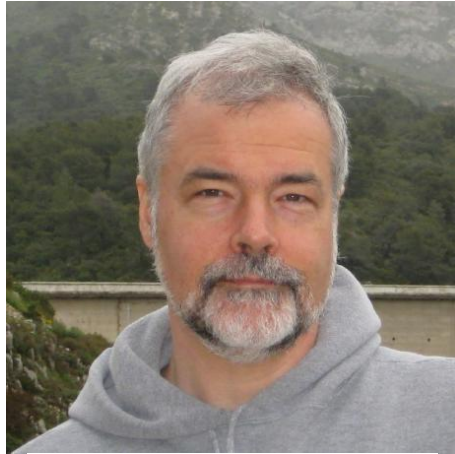
Use additional sheets to produce CLI commands

Really, Cost Explorer is your BFF



Questions?

About Me



AWS Practice Lead at Chariot Solutions

Programming since 1977,
professionally since 1984,
on AWS since 2008

<https://www.kdgregory.com/>

<https://github.com/kdgregory/>

@ChariotKGregory





Technology in the Service of Business.

Chariot Solutions is the Greater Philadelphia region's top IT consulting firm specializing in software development, systems integration, mobile application development and training.

Our team includes many of the top software architects in the area, with deep technical expertise, industry knowledge and a genuine passion for software development.

Visit us online at chariotsolutions.com.

